TEKNILLINEN KORKEAKOULU
Tietotekniikan osasto

# Automaattinen hajautetun palvelinympäristön valvontajärjestelmä

Harri Pylkkänen

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Lahdessa 15.5.2005
Työn valvoja: prof. Juha Tuominen

| TEKNILLINEN KORKEAKOULU<br>Tietotekniikan osasto | | | DIPLOMITYÖN TIIVISTELMÄ |
|---|---|---|---|
| Tekijä | Harri Pylkkänen | | |
| Työn nimi | Automaattinen hajautetun palvelinympäristön valvontajärjestelmä | | |
| Päivämäärä | 15.5.2005 | Sivumäärä | 65 |
| Professuuri | Vuorovaikutteinen digitaalinen media | Koodi | T-111 |
| Valvoja | Professori Juha Tuominen | Ohjaaja | - |


Tietotekniikan hyödyntäminen yhteiskunnan eri osa-alueilla on kasvanut viimeisten vuosikymmenien aikana huomattavasti. Toimintoja pyritään tehostamaan tietotekniikan suomin keinoin ja optimoimaan teknisin innovaatioin. Tietotekniikan käytön lisääntymisen myötä on jouduttu tilanteeseen, jossa tietojärjestelmien toimivuus on elinehto kokonaisuuden toimivuudelle.

Tietojärjestelmän toimivuudesta vastaa tämän ylläpito, jonka tehtävänä on huoltaa, korjata ja parantaa tietojärjestelmää. Näillä edesottamuksilla pyritään tilanteeseen, jossa tietojärjestelmä olisi aina käytettävissä, saatavilla. Tietojärjestelmien koon ja merkityksen kasvaessa pyritään tietojärjestelmän toimintavarmuutta parantamaan monistamalla tietojärjestelmän kriittisiä komponentteja ja hajauttamalla toimivia kokonaisuuksia tämän mahdollistamiseksi. Tietojärjestelmien laajentuessa myös näiden ylläpito muodostuu haasteellisemmaksi valvottavan tiedon hajautuessa laajalle alueelle ja kriittisten tietojen jäädessä informaatiotulvan peittoon.

Työn yleisessä osassa perehdytään ylläpidon taustoihin, ylläpidon erilaisiin sopimuksiin sekä ylläpidon luokitteluun korjaavaan, mukauttavaan, kehittävään ja ennaltaehkäisevään ylläpitoon. Ennaltaehkäisevään ylläpitoon paneudutaan syvällisemmin esittäen asioita, joiden avulla ongelmatilanteisiin voidaan varautua ja kuinka valvonta kuuluu ennaltaehkäisevän ylläpidon kantavaksi voimaksi.

Työn erityisessä osassa pyritään määrittelemään geneerinen valvontasovellus, jonka avulla voidaan hajautetun tietojärjestelmän valvontatieto koostaa ylläpidon kannalta helpommin hallittavaksi siten, että valvontajärjestelmä kykenee keräämään valvontatietoa myös vioittuneesta tietojärjestelmästä.



Avainsanat
hajautettu tietojärjestelmä, tietojärjestelmän ylläpito, ennaltaehkäisevä ylläpito, tietojärjestelmän saatavuus, tietojärjestelmän valvonta, Linux, Java

| HELSINKI UNIVERSITY OF TECHNOLOGY<br>Department of Computer Science and Engineering | | ABSTRACT OF<br>MASTER'S THESIS | |
|---|---|---|---|
| Author | Harri Pylkkänen | | |
| Title of thesis | Automatical survaillance software for distributed server environment | | |
| Date | 15.5.2005 | Number of pages | 65 |
| Professorship | Interactive digital media | Professorship code | T-111 |
| Supervisor | Professor Juha Tuominen | Instructor | - |


Usage of computer technology in different part of society has increased noticeably during last decades. Procedures are being boosted by methods of information technology and optimized by technical innovations. The increased usage of information technology has put us to a situation, where functioning information systems are prerequisite for the wholeness to work.

Maintenance team is responsible for keeping information systems in a working state by servicing, repairing and upgrading it. With these doings are being aimed at to a situation, where information system is always usable and available. When size and importance of information systems is increasing their reliability are being improved by multiplying critical components of the system and by distributing functioning entities to make this possible. When information systems are expanding their maintenance is getting more demanding, because information to be survailled is distributed to wide area and critical information is left behind the information flow.

In the general part of the work the background of maintenance is being explained by cathegorizing it to repairing, adapting, upgrading and preventive maintenance. Also different kind of agreements of maintenance are being viewed. Preventive maintenance is viewed more thoroughly by presenting ways to prepare for problem situations and by showing how surveillance is one of the leading forces of it.

As own work a generic surveillance software is being defined. With this software the surveillance information of distributed information system can be presented in a more compact way to lighten the maintenance work. The surveillance software can gather information also from injured information system.



Keywords
distributed information system, maintenance of information system, preventive maintenance, availability of information system, surveillance of information system, Linux, Java

# Alkulause

Tämän diplomityön toteutuksen todellisena haasteena oli toteutuskelpoisen aiheen löytäminen siten, että tämä olisi tieteellisesti riittävän haastava ja muutoin motivoiva. Idean valvontasovelluksen toteuttamiseen löytyi esimieheltäni, Johannes Lehtiseltä, josta minun tulee olla kiitollinen. Myös Satu Schaeffer auttoi tässä vaikeassa vaiheessa punnitsemaan mahdollisia vaihtoehtoja, josta hänellekin kuuluu suuri kunnia. Työn toteuttamisessa professori Juha Tuomisen neuvot ovat tulleet tarpeeseen ja ovat esimerkillisesti ohjanneet kohti päämäärää.

Työni toteuttamisen mahdollistamiseksi tahdon kiittää Urho ja Kaisu Kiukkaan säätiötä sekä Hilja ja Alpo Savolaisen rahastoa heidän myöntämistään stipendeistä ja nykyistä työnantajaani, Rossum Oy:a, jotka ovat edesauttaneet minua vastaanottamaan ne haasteet, joita opiskelu Teknillisessä Korkeakoulussa ja sieltä valmistuminen vaatii.

Kirjoitusprosessin läpiviemisessä rakkaat ystäväni ovat olleet tukemassa minua ihailtavasti. He ovat piristäneet minua silloin, kun työ ei ole näyttänyt edistyvän ja ovat luoneet uskoa esteiden ylittämisen mahdollisuuksiin. Muistan teitä kaikkia lämmöllä.

Lahdessa, 15.5.2005

Harri Pylkkänen

# Sisältö


**1 Johdanto** **1**

1.1 Tietojärjestelmien yleistyminen . . . . . 1
1.2 Tietojärjestelmien toimivuuden takaaminen . . . . . 3
1.3 Tavoitteet . . . . . 3
1.4 Lukuohje . . . . . 3

**2 Tietojärjestelmän ylläpito** **5**

2.1 Historiaa . . . . . 6
2.2 Taustaa . . . . . 6
2.3 Ylläpidon merkityksen muuttuminen . . . . . 7
2.3.1 Käyttäjä asiantuntijana . . . . . 8
2.3.2 Ylläpidon vastuun siirtyminen käyttäjältä ylläpitoryhmälle 8
2.3.3 Toimintavarmuuden arvostus kasvaa . . . . . 8
2.4 Ylläpitosopimukset . . . . . 9
2.4.1 Huoltosopimus . . . . . 9
2.4.2 Ohjelmiston ylläpitosopimus . . . . . 10
2.4.3 Tukisopimus . . . . . 10
2.5 Palvelusopimukset . . . . . 10
2.5.1 Käyttöpalvelu . . . . . 11
2.5.2 Sovellusvuokraus . . . . . 11
2.5.3 Palvelutasoliite . . . . . 11
2.6 Asiakaspalvelu . . . . . 12
2.7 Korjaava ylläpito . . . . . 12
2.8 Mukauttava ylläpito . . . . . 13
2.9 Kehittävä ylläpito . . . . . 13
2.10 Ennaltaehkäisevä ylläpito . . . . . 14
2.11 Yhteenveto . . . . . 14

**3 Ennaltaehkäisevä ylläpito** **15**

3.1 Saatavuuden takaaminen . . . . . 16
3.1.1 Ongelmatilanteiden välttäminen . . . . . 17
3.1.2 Ongelmatilanteisiin varautuminen . . . . . 18
3.1.3 Ongelmatilanteiden selvittäminen . . . . . 18
3.2 Palvelun kehittäminen . . . . . 19
3.3 Laitealustan valvonta . . . . . 19

3.4 Ohjelmistoalustan valvonta . . . . . 21
3.5 Hajautetun ohjelmisto- ja laitealustan valvonta . . . . . 21
3.6 Huolto . . . . . 23
3.7 Päivitykset . . . . . 23
3.8 Tietoturva . . . . . 24
3.8.1 Ennaltaehkäisy . . . . . 25
3.8.2 Havaitseminen . . . . . 25
3.8.3 Rajoittaminen . . . . . 25
3.8.4 Palautuminen . . . . . 25
3.8.5 Korjaaminen . . . . . 25
3.9 Yhteenveto . . . . . 25

**4 Hajautetun ohjelmisto- ja laitealustan valvonta 27**
4.1 Taustaa . . . . . 28
4.1.1 Hajautusteknologiat . . . . . 28
4.1.2 Toteutusmallit . . . . . 28
4.1.3 Väliohjelmistot . . . . . 28
4.1.4 Valvonta . . . . . 29
4.2 Tavoitteet . . . . . 29
4.3 Kaupallinen motivaatio . . . . . 30
4.4 Edeltävä tilanne . . . . . 31
4.5 Yhteenveto . . . . . 33

**5 Automaattisen valvontasovelluksen määrittely 34**
5.1 Tiedonsiirto ja valvontaverkoston rakenne . . . . . 34
5.1.1 Tilatiedon hankkiminen alikomponentilta . . . . . 36
5.2 Tietojen reaaliaikaisuus . . . . . 37
5.3 Tietokannan toteutus . . . . . 37
5.4 Toimivuuden takaaminen . . . . . 39
5.4.1 Alimman tason komponentin valvontatietojen hankinta epäonnistuu . . . . . 41
5.4.2 Komponentin informaatiota ei olla tiedusteltu tietyn aikarajan puitteissa . . . . . 41
5.4.3 Hälytysviestejä ei pystytä lähettämään ylläpidolle . . . . . 42
5.4.4 Menettely valvontaverkon ongelmatilanteessa . . . . . 42
5.5 Tulevien ongelmatilanteiden ennustaminen . . . . . 44
5.6 Ohjelman rakenne . . . . . 45
5.7 Tiedonvälitysrajapinta . . . . . 45
5.7.1 Tietojen pyyntisanoma . . . . . 46
5.7.2 Tietojen välityssanoma . . . . . 46
5.7.3 Tietojen vastaanottamisen kuittaussanoma . . . . . 48
5.7.4 Konfiguraation muutospyyntösanoma . . . . . 48
5.7.5 Konfiguraation muutoksen vahvistussanoma . . . . . 49
5.7.6 Konfiguraation muutosilmoitussanoma . . . . . 50
5.8 Tiedonvälityksen turvaaminen . . . . . 50

5.9 Yhteenveto . . . . . 51

**6 Automaattisen valvontasovelluksen toteutus** **52**
6.1 Ohjelmointikielen päättäminen . . . . . 52
6.1.1 Java . . . . . 53
6.1.2 Komentotulkkiskriptit sovelluksen osana . . . . . 53
6.2 Valvontaverkon sanomaliikenteen salaus . . . . . 54
6.3 Ohjelmistototeutukseen tehdyt rajaukset . . . . . 54
6.3.1 Tietokanta . . . . . 54
6.3.2 Käyttöliittymä . . . . . 54
6.3.3 Vikatilanteesta palautuminen normaalitilaan . . . . . 55
6.3.4 Ongelmatilanteiden ennustaminen . . . . . 55
6.4 Määritelmiin tehdyt muutokset . . . . . 55
6.5 Yhteenveto . . . . . 56

**7 Testaus** **57**
7.1 Moduulitestaus . . . . . 57
7.1.1 Valvontatiedon lukeminen laitealustalta . . . . . 58
7.1.2 Valvontatiedon pyytäminen toiselta komponentilta . . . . 58
7.1.3 Valvontatiedon lähettäminen toiselle komponentille . . . . 58
7.1.4 Sanomaliikenteen salaus . . . . . 58
7.1.5 Sanomaliikenteen salauksen purku . . . . . 58
7.1.6 Komponentin tiedon esitys käyttöliittymässä . . . . . 58
7.2 Integraatiotestaus . . . . . 59
7.2.1 Tiedon vastaanottaminen toiselta komponentilta . . . . . 59
7.3 Järjestelmätestaus . . . . . 59
7.3.1 Valvottavan järjestelmän tilatietojen esittäminen . . . . . 59
7.3.2 Vikatilanteen havaitseminen järjestelmän tilatiedoista . . 59
7.3.3 Vikatilanteen havaitseminen valvottavassa aliverkossa . . 60
7.3.4 Vikatilanteen havaitseminen ylemmän tason komponentissa . . . . . 60
7.4 Yhteenveto . . . . . 60

**8 Testauksen tarkastelu** **61**
8.1 Sovelluksen ongelmakohdat . . . . . 61
8.1.1 Komponenttien dynaamisen lisäämisen puuttuminen . . . 61
8.1.2 Hälytysrajojen konfiguroinnin hankaluus . . . . . 61
8.1.3 Laitealustan vaikutus suorituskykyyn . . . . . 62

**9 Yhteenveto** **63**

**10 Tulevaisuus** **65**

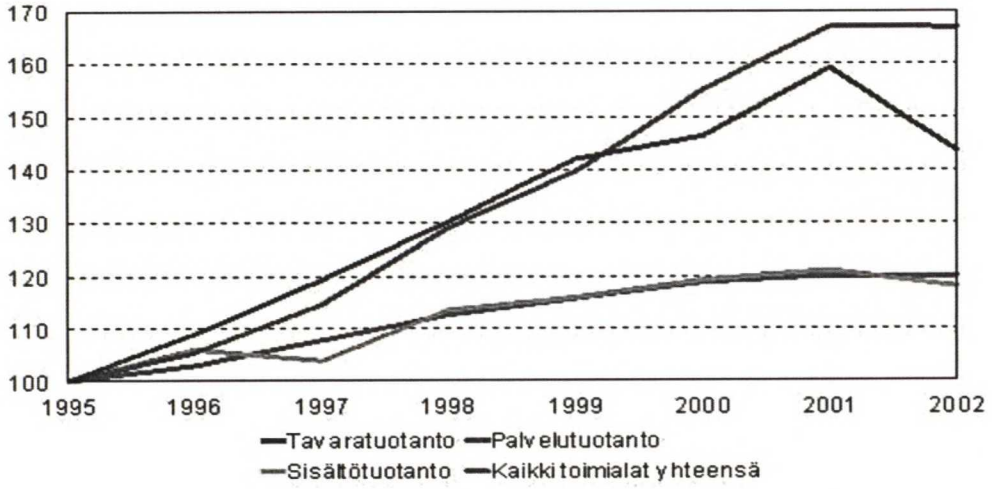
# Luku 1

# Johdanto

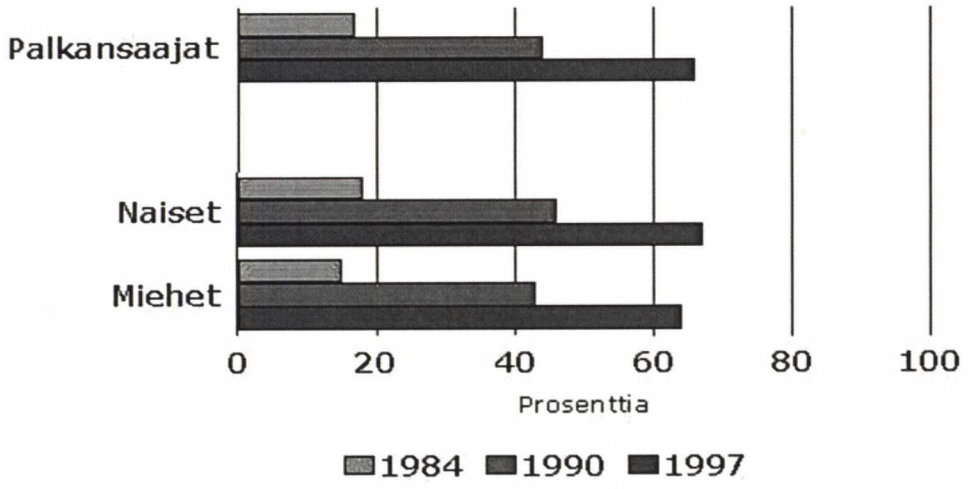
Viimeisimpien vuosikymmenien kehityksen tuloksena tiedosta on tullut yhä keskeisempi osa kaupankäyntiä ja tämän laaja hyväksikäyttö sekä tieto- ja viestintäteknologian hyödyntäminen muun muassa kilpailukyvyn ja hyvinvoinnin edistämiseksi on muodostunut oleelliseksi osaksi jokapäiväistä kanssakäyntiämme. "Yhteiskunnan kehittyessä muuttuvat tuotanto, työelämä, koulutus, jakelukanavat, kulutustottumukset ja jokapäiväinen elämä", todetaan Tilastokeskuksen Internet-sivustolla tietoyhteiskuntaa käsittelevässä osiossa [18]. Yhteiskunnastamme on muodostunut tietoyhteiskunta.

## 1.1 Tietojärjestelmien yleistyminen

Suomen elinkeinorakenne on muuttunut viimeisen vuosikymmenen aikana entistä palveluvoittoisemmaksi ja tietoteknisten alojen suosio on kasvanut kaiken aikaa. Tällaiseen kehitykseen vaikuttavat niin teknologian kehitys, palvelujen kasvava merkitys kuin kansainvälistyminen ja väestön ikärakenne. Yritystoiminnassa informaatiosektori, joka sisältää tavara-, palvelu- ja sisällöntuotannon toimialoja, on kehittynyt huimaa vauhtia, sillä vuosien 1995 ja 2001 välisenä aikana sektoriin kuuluvien yritysten liikevaihto kolminkertaistui ja henkilöstömäärä lisääntyi lähes 50 prosenttia. Tämä ilmenee Tilastokeskuksen kokoomajulkaisusta Tiedolla tietoyhteiskuntaan IV [17]. Henkilöstömäärän kehitystä informaatiosektorilla kuvaa oheinen kuvaaja 1.1. Myös tietotekniikan yleistyminen yrityksissä on ollut voimakasta viimeisten parinkymmenen vuoden aikana, sillä jo vuonna 1997 noin 65 prosenttia palkkatyöläisistä hyödynsivät tietotekniikkaa, kun vuonna 1984 vastaava luku oli noin 17 prosenttia ja kehityksen suunta on enenevissä määrin kohti tietokoneistuneempaa tulevaisuutta. Tietotekniikan käytön kehitystä työntekijöiden kannalta kuvaa kuvaaja 1.2.




Kuva 1.1: Henkilöstömäärän kehitys 1995-2002 informaatiosektorilla [18].




Kuva 1.2: Tietotekniikan yleistyminen palkansaajien keskuudessa 1984-1997 [18].

Tietokoneiden kasvanut kyky prosessoida suuria tietomääriä kerralla soveltuu kasvaneiden tietomäärien käsittelyyn ja kehittynyt telekommunikaatio laajentaa tämän globaaliksi kokonaisuudeksi. Tietotekniikkaa hyödynnetään tänä päivänä useissa eri sovelluskohteissa tehostamaan tai mahdollistamaan tehtävien suorittamista. Tällaisia tietoteknisiä kokonaisuuksia kutsutaan tietojärjestelmiksi.

Yritysten kannalta tietotekniikka on luonut mahdollisuuden optimoida tehtäviä ja tehdä niistä näinollen kustannustehokkaampia. Tämä on puolestaan johtanut useat organisaatiot riippuvaisiksi tietojärjestelmistä, sillä näissä ilmenneet ongelmat aiheuttavat ylitsepääsemättömiä vaikeuksia toiminnan jatkamiselle, koska tuotanto- ja hallintaprosessit on optimoitu hyödyntämään tietojärjestelmien suomaa prosessointikykyä.

## 1.2 Tietojärjestelmien toimivuuden takaaminen

Tietoyhteiskunnan yksi perusedellytyksistä on varmistua siitä, että sen mahdollistavat tietotekniset järjestelmät pysyvät toimintakuntoisina mahdollisimman suuren osan ajasta, jotta vältyttäisiin näistä järjestelmistä riippuvaisien operaatioiden estymiseltä. Tämä korostaa tietojärjestelmien ylläpidon merkitystä niin julkisella kuin yksityiselläkin sektorilla sillä tietojärjestelmien on suoriuduttava niille asetetuista tehtävistä tietyn ajan sisällä ja tämän takaamiseksi tulee varata riittävästi resursseja valvomaan järjestelmien toimintaa ja tarvittaessa huoltamaan vikaantunutta järjestelmää.

## 1.3 Tavoitteet

Tämän lopputyön tavoitteena on tutustua tietojärjestelmien ylläpidon eri osa-alueisiin ja ylläpitoon liittyviin erilaisiin sopimuksiin ylläpitävän tahon ja asiakkaan välillä. Erityisesti hajautettujen tietojärjestelmien ylläpitoon liittyvät ongelmat otetaan esiin.

Lopputyön oman työn osuudessa tavoitteena on pyrkiä toteuttamaan ohjelmisto, jota voidaan käyttää hajautetun tietojärjestelmän tilan valvontaan. Tämän valvontaohjelmiston tulisi pystyä keräämään tietoa eri tietojärjestelmän komponenttien tilasta ja määriteltyjen kriittisten raja-arvojen ylityttyä ilmoittaa tietojärjestelmän ylläpidolle mahdollisesta ongelmatilanteesta. Hajautettua tietojärjestelmää valvovan ohjelmiston tulee pystyä selviytymään tiedonsiirtoyhteyksissä esiintyvistä ongelmista ja tietojärjestelmän osittaisista vikatilanteista siten, että ylläpidolle välittyy ajantasalla olevaa tietoa näistä ongelmallisista alueista. Valvotun tietojärjestelmän tilaa tulee voida seurata myös pidemmältä aikaväliltä, jolloin mahdollisia tietojärjestelmän käyttäytymismalleja kyetään löytämään. Toteutettavan ohjelmiston tarkoituksena on toisin sanoen yhtenäistää hajautetusta tietojärjestelmästä saatavaa tietoa, kohdentaa huomiota mahdollisiin tietojärjestelmän ongelmakohtiin ja tätä kautta tehostaa ja selkeyttää tietojärjestelmän ylläpitoa.

## 1.4 Lukuohje

Luvussa 2 esitellään tietojärjestelmän ylläpidon merkityksen muuttumista tietojärjestelmien yleistymisen myötä ja erilaisia toimittajan ja asiakkaan välisiä sopimuksia, joiden tarkoitus on luoda selkeät toimintamallit kyseessä olevaan tietojärjestelmään kohdistuvien tilanteiden varalle. Luvussa esitellään myös ylläpitokäsitteeseen kuuluvia kategorioita.

Luvussa 3 käsitellään tarkemmin ennaltaehkäisevää ylläpitoa eli toimintoja, joilla pyritään ehkäisemään ongelmatilanteet järjestelmässä ennen niiden tapahtumista. Luvussa käsitellään tietojärjestelmän saatavuutta ja siihen vaikuttavia tekijöitä. Muita käsiteltäviä aiheita ovat muun muassa tietojärjestelmän päivittäminen ja huoltaminen.

Luvussa 4 käsitellään hajautetun järjestelmän rakennetta ja edetään lopputyön oman työn osuuteen esittelemällä tutkimuksen kaupallisia tavoitteita, aiempaa järjestelmää ja tarkennetaan lopputyönä toteutettavan valvontasovelluksen tavoitteita.

Luvussa 5 määritellään toteutettavan valvontasovelluksen rakennetta, logiikkaa ja sanoman välitystä.

Luvussa 6 käsitellään valvontasovelluksen toteutusvaiheita, tämän aikana tehtyjä havaintoja ja muutoksia.

Luvussa 7 perehdytään valvontasovelluksen testauksessa käytettyihin menetelmiin ja itse testien suorittamiseen.

Luvussa 8 luodaan katsaus valvontasovelluksen testituloksiin ja selvitetään niissä tehtyjä havaintoja.

Luku 9 sisältää pohdintaa lopputyöprojektin onnistumisesta, saavutuksista ja tehdyistä havainnoista kokonaisuutena.

Luku 10 sisältää katsauksen ylläpidon automatisoinnin tulevaisuuden näkymiä sekä mahdollisia kehityssuuntia, joihin tutkimuskohdetta olisi mahdollista kehittää.

# Luku 2

# Tietojärjestelmän ylläpito

Asenteet tietojärjestelmien toteutukseen ovat muuttuneet vuosikymmenten aikana paljon. Kuten Heikki Koistinen asian esittää kuvatessaan tietojärjestelmien toteutustapoja[9], ovat yritykset 1960-luvun itsetoteutetuista tietojärjestelmistä eriyttäneet kehitystä ulkoisille tahoille pyrkien kohdistamaan oman panoksensa ydinosaamisalueeseensa ja antaneet näin mahdollisuuden valmisohjelmistojen ja -ratkaisujen nousulle ohjelmistoteollisuuden suurimmaksi osa-alueeksi. Kasvavana suuntauksena on viime vuosina ollut sovellusvuokraustoiminta, jossa yritys hankkii haluttuun ohjelmistoon ainoastaan käyttöoikeudet ja itse ohjelmistoa operoidaan sovellusta vuokraavan yrityksen tietojärjestelmässä.

Erityisesti sovellusvuokraustoiminnassa on olennaista, että tietojärjestelmä, jossa ohjelmistoa ajetaan, on hyvin ylläpidetty ja vakaa, sillä riippuvathan palvelua tarjoavan yrityksen tulot käytön määrästä, kuten Pekka Takki huomauttaa sovellusvuokrauksesta[16]. Asiakkaan maksaessa ainoastaan käytön mukaan, maksaakin hän aiemmasta poiketen saamastaan palvelusta, eikä ohjelmistosta itsestään.

Asiakkaan konkreettisten investointien määrän vähentyessä siirryttäessä uuden tietojärjestelmän käyttäjäksi heikentää asiakkaan ja tuottajan välistä suhdetta ja tällöin kilpailevien yritysten on yhä helpompaa saada kyseinen asiakas itselleen laadukkaamman tuotteen, paremman palvelun tai edullisemman hintatason avulla. Tuottajalle on tällaisessa asiakkuussuhteessa paljon tärkeämpää pitää tietojärjestelmänsä toimintakuntoisena, sillä näin he luovat itsestään parempaa kuvaa potentiaalisille asiakkaille ja pitävät olemassa olevat asiakkaansa tyytyväisinä.

Tämän luvun tarkoituksena on tutustuttaa tietojärjestelmien taustaan, toimittajan ja asiakkaan välisiin ylläpito- ja palvelusopimuksiin sekä erilaisiin ylläpidon tyyppeihin. On oleellista havaita ylläpidon laaja-alaisuus ja suuri toimijoiden joukko, jotka välillisesti tai välittömästi vaikuttavat toimivan tietojärjestelmän aikaansaamiseen.

## 2.1 Historiaa

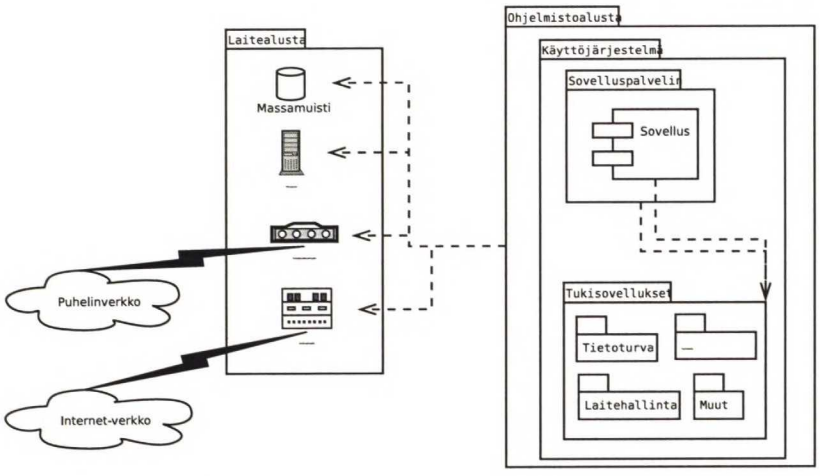
Tietojärjestelmät ohjelmiston tukirakenteena ovat kasvattaneet merkitystään ohjelmistokulttuurin muuttuessa vuosikymmenten varrella. Aluksi yritykset tekivät ohjelmistot alusta alkaen itse, jolloin ohjelmisto muovautui vastaamaan tietojärjestelmän suomaa rajapintaa ja havaitut ongelmakohdat pystyttiin korjaamaan joko tietojärjestelmään tai itse ohjelmistoon. Tällaista menettelytapaa sovellettiin aina 1990-luvulle asti, jolloin tällaisten ratkaisujen ylläpitämiset todettiin vievän yritysten fokusta pois sen ydinosaamisesta.

1970-luvulta lähtien kasvavana suuntauksena ohjelmistojen toteutusta ulkoistettiin, joko ostamalla ohjelmisto valmiina tai räätälöimällä ohjelmisto yrityksen omiin tarpeisiin sopivaksi. Havaittavaa tällaisessa menettelyssä on se, että ohjelmistot toimivat silti asiakasyrityksen tietojärjestelmässä, jolloin ohjelmiston ja tietojärjestelmän ylläpito lankeaa hyvin todennäköisesti asiakasyrityksen vastuulle.

2000-luvulla kiinnostus ohjelmiston suoman palvelun ostamisesta ulkopuoliselta taholta kasvoi ja syntyi käsite sovellusvuokraus, jolloin asiakasyrityksellä ei ole hallussa ohjelmistoa eikä tähän liittyvää tietojärjestelmää. Asiakasyrityksen ei tällaisella menettelyllä tarvitse huolehtia ohjelmiston ylläpidosta ollenkaan, sillä ostaessaan pelkästään ohjelmiston suomaa palvelua, on ylläpito täysin ohjelmistoa tarjoavan yrityksen vastuulla. Ylläpidon merkitys muuttuu tällaisessa tilanteessa asiakasyrityksen toimintaa tukevasta toiminnasta elinehdoksi ohjelmistoa tarjoavan yrityksen toiminnalle. Tätä ohjelmistokulttuurin muutosta ja järjestelmien toteutuksen kehitystä kuvaa teoksessaan Heikki Koistinen [9].

## 2.2 Taustaa

"Tietojärjestelmä on ihmisistä, tietojenkäsittelylaitteista, tiedonsiirtolaitteista ja ohjelmista koostuva järjestelmä, jonka tarkoitus on tietoja käsittelemällä tehostaa tai helpottaa jotakin toimintaa tai tehdä toiminta mahdolliseksi", todetaan Valtionvarainministeriön tietoturvasanastossa [20]. Konkreettisesti tietojärjestelmä voidaan jaotella fyysiseen osaan palvelimineen ja tietoverkkoineen ja ohjelmisto-osuuteen käyttöjärjestelmineen, tietokantoineen ja muine tukisovelluksineen, kuten Hairo selventää lopputyössään [6]. Tietojärjestelmän rakennetta on havainnoillistettu kuvassa 2.1. Tietojärjestelmään vaikuttavat ihmiset niin operoijina kuin ylläpitäjinäkin on jätetty huomiotta kuvassa.




Kuva 2.1: Tietojärjestelmän havainnekuva

Laitealusta, tietokone, on massa- ja käyttömuisteineen, prosessoreineen ja mahdollisine verkkoyhteyksineen toteutettu suoriutumaan sille osoitetuista prosesseista siten, että lopputulos suurenkin käyttöasteen alla on odotetun kaltainen ja että se pystyy tekemään tätä vakaasti pitkiä aikajaksoja kerrallaan.

Tietojen jakaminen erilaisten järjestelmien kesken ja internetin suosion kasvaessa on tietojärjestelmien tiedonsiirtoyhteyksien merkitys kasvanut kahden viime vuosikymmenen aikana merkittävästi. Tietojen välittämisellä järjestelmältä tai sen osalta toiselle saadaan luotua kokonaisvaltaisempi kuva vallitsevasta tilanteesta ja tietojärjestelmää hyödyntävät ohjelmistot pystyvät näin tekemään työnsä tehokkaammin ja luomaan laajemman näkemyksen käsiteltävästä ongelmasta.

Laitealusta itsessään tarvitsee suoriutuakseen saamistaan komennoista joukon tukiohjelmia, jotka muuttavat saadut ohjelmakomennot laitealustan ymmärtämään muotoon ja palauttavat prosessoitujen komentojen tulokset takaisin herätteen tehneille ohjelmille. Samanlaisessa asemassa ovat myös ohjelmistoalustat yksinkertaistaessaan laitealustaan kohdistuvat rajapinnat ja samalla tehden suoritettavasta ohjelmistosta laitealustariippumattomamman. Ohjelmistoalustat yksinkertaistavat uusien ohjelmien toteutusta, sillä näin pystytään ohjelmiston toteutuksessa keskittymään itse ongelman ratkaisemiseen, kun laitteistorajapinnat on hoidettu ohjelmistoalustan puolesta.

## 2.3 Ylläpidon merkityksen muuttuminen

Tietotekniikan yleistyessä sen käyttäjäkunta on laajentunut asiantuntijoista tavallisiin kansalaisiin. Tämä on muovannut myös tietojärjestelmien ylläpitokäsitettä huomattavasti, koska loppukäyttäjiltä ei voida enää tänä päivänä odottaa tietojärjestelmien laaja-alaista tuntemusta. Asiantuntijoiden käyttämistä järjestelmistä, joita he pääosin itse ylläpitivät, on siirrytty koulutetun

käyttäjäkunnan sovelluksiin, joiden toimintakunnosta vastaavat erityiset ylläpitoryhmät.

### 2.3.1 Käyttäjä asiantuntijana

Vuodelta 1982 peräisin oleva tietoliikennealan ohjelmistojen ylläpitoa käsittelevä tiedote [11] määrittää kehittäjän näkökulmasta ylläpidon osiksi virheiden korjauksen, sisäisen parantamisen, sovittamisen, toimintojen lisäämisen ja puhelinteknisen laajentamisen ja ohjelmiston käyttäjän osalle jäävät järjestelmän datamuutokset, valvonta ja vikatilanteiden poistaminen.

Erityisesti järjestelmän käyttäjän vastuulle jätetään suuri vastuu valvonnan suorittamisesta ja datamuutosten suorittamisesta järjestelmään. Voidaan havaita, että 80-luvun alussa tietojärjestelmät oli suunnattu asiantuntijoiden käyttöön, joiden oletettiin tuntevan järjestelmän käyttäytyminen ja kykenemään toimimaan havaituissa ongelmatilanteissa.

### 2.3.2 Ylläpidon vastuun siirtyminen käyttäjältä ylläpitoryhmälle

Tietojärjestelmien käyttökohteiden monipuolistuminen puhtaasta tiedonkäsittelystä yksinkertaisempiin tiedon talletus- ja käsittelysovelmiin ja yrityskohtaisiin sovellusalueisiin muokkasi myös tietojärjestelmien käyttäjäkuntaa. Tietojärjestelmien käyttämistä yksinkertaistettiin siinä määrin, että tavallisen käyttäjän ei enää tarvinnut tuntea järjestelmän kaikkia ominaisuuksia, vaan hän pystyi suorittamaan operaationsa jo lyhyelläkin koulutuksella. Ongelmatilanteiden varalta tietojärjestelmien kunnosta huolehtivat järjestelmän toiminnan tuntevat ylläpitäjät.

Suomalaisten tietojärjestelmien ylläpidosta vuonna 1988 teetetty tutkimuksessa [23] ylläpidon määritettiin koostuvan tietojärjestelmän kehittämisestä ja kunnossapidosta. Kunnossapito käsitti tuolloin virheiden korjaamisen eli korjaavan ylläpidon sekä toimintakunnon säilyttämisen eli mukauttavan sekä huoltavan ylläpidon.

Tavallisen käyttäjän vastuulta poistui tässä vaiheessa näinollen asiantuntijajärjestelmien vikatilanteiden selvitykset, valvonnat sekä datamuutokset ja nämä siirrettiin ylläpidosta vastaavan tahon harteille.

### 2.3.3 Toimintavarmuuden arvostus kasvaa

Yritysten automaatiotason noustessa ja tietojärjestelmien määrän kasvaessa on ylläpidon rooli muuttunut entisestään. Kuten Kaikkonen lopputyössään selventää [7], koostuu ylläpito korjaavasta ja ennaltaehkäisevästä ylläpidosta. Korjaava ylläpito kattaa tilanteet, jossa järjestelmä on syystä tai toisesta estynyt toimimasta ja vaatii ylläpidollisia toimia toiminnan normalisoitumiseksi. Ennaltaehkäisevä ylläpito puolestaan sisältää luotettavuutta kehittävää toimintaa sekä toimia suunnittelemattomien ylläpitotoimien minimoimiseksi. Kaikkonen huomauttaa työssään, että oleellista on löytää ylläpidon

osalta kustannustehokas tasapaino korjaavan ylläpidon ja ennaltaehkäisevän ylläpidon välille.

## 2.4 Ylläpitosopimukset

Tietojärjestelmää hankkiessaan asiakas haluaa kokonaisuudelle mahdollisimman kattavan aktiivisen ja passiivisen tuen, mutta tämä ei yleisesti ottaen ole mahdollista, sillä toimittajalla tai toimittajien joukolla on rajalliset resurssit järjestää tukea ja kustannusten nousu aktiivisemman tuen myötä ajaa neuvottelevat osapuolet hakemaan kompromisseja.

Passiivinen tuki voidaan järjestelmän kohdalla mieltää käyttöohjeeksi ja järjestelmässä toimivan ohjelmiston osalta siihen liitettyyn aputoimintoon, joihin on listattu yleisimmät ongelmatilanteet ja ohjeet siitä kuinka näistä ongelmista pystyy selviytymään omatoimisesti.

Aktiivinen tuki puolestaan tarkoittaa, että järjestelmän käyttäjä voi kysyä apua toimittajan tai hallinnoijan osoittamasta kontaktikanavasta ja että järjestelmän tilaa seurataan aktiivisesti ja siinä havaittuja ongelmia ryhdytään korjaamaan kohtuullisessa ajassa.

Sopimuksissa määritellään, kuinka nopeasti tukipyyntöihin reagoidaan, mihin aikaan vuorokaudesta tukea on saatavilla, millaiset viat kuuluvat takuun piiriin ja millaisella hinnastolla mahdollisia muutostöitä tuotteelle tehdään. Toimittajan kannalta pitkäkestoiset sopimukset asiakkaan kanssa luovat toiminnalle varmuutta ja motivoivat toimittajaa panostamaan ylläpitoon, kun taas asiakkaalle pitkät sopimuskaudet sitovat mahdollisesti asiakasta toimittajan järjestelmiin. Sopimuksilla määritellään kumpaakin osapuolta tyydyttävä ratkaisu halutusta tuesta ja kustannusten määräytymisestä.

Selkeyden vuoksi ylläpitopalvelut ryhmitellään yleensä laitteiston tilan seurantaan, ohjelmiston ylläpitopalveluun ja käyttäjätukeen. Näiden sisältöä käsitellään sopimusteknisesti vastaavasti huoltosopimuksessa, ohjelmiston ylläpitosopimuksessa ja tukisopimuksessa.

### 2.4.1 Huoltosopimus

Huoltosopimus pitää sisällään järjestelmän laitealustan mekaanisten vikojen ja toimintahäiriöiden korjaamisen ja laitealustan säätämisen kaltaisia toimenpiteitä eli jos laitealustassa ilmenee ongelmia, toimittaja paikantaa vian ja korjaa tämän. Sopimuksessa määritetään palvelun hinnoittelukäytäntö eli määritetään palvelun perusmaksu, huoltotöiden hinnoittelu ja varaosien veloittamiskäytäntö. Pekka Takin mukaan on tärkeää havaita, että huoltosopimus itsessään aikaansaa toimittajalle ainoastaan toimintavelvoitteen, eikä tulosvelvoitetta, jota asiakas yleensä toimittajalta ongelmatilanteessa odottaa [16]. Tämä tarkoittaa sitä, että asiakas olettaa toimittajan tekevän huoltotyöt mahdollisimman ripeästi, vaikka sopimusehto täyttyy toimittajan kannalta jo siinä, että toimittaja ryhtyy korjaamaan laitteistoa.

Tällaisten huoltotöiden tehokkuuskiistojen välttämiseksi huoltosopimukseen lisätään yleensä myös toimittajan toiminnalle vasteaika, jonka puitteissa toimittajan on reagoitava asiakkaan vikailmoitukseen. Huoltosopimus voi myös sisältää käytettävyyssitoumuksen, jossa toimittaja vastaa siitä, että laitteisto on käytettävissä tietyn prosentuaalisen osuuden asiakkaan kuukausittaisesta työajasta. Toimittajaa velvoittavien sopimuskohtien täyttämättä jättäminen tai tavoiteaikojen laiminlyömiset hyvitetään asiakkaalle yleensä alentamalla kuukausittaista ylläpitomaksua riippuen havaittujen ongelmakohtien vakavuudesta.

### 2.4.2 Ohjelmiston ylläpitosopimus

Tietojärjestelmän ohjelmiston ylläpitosopimuksen laatiminen on huoltosopimuksen laatimiseen verrattaessa monimutkaisempaa, sillä ohjelmiston ylläpidollisen sisällön määrittäminen on epämääräisempää ja mitään konkreettista määrittelyä on hankala työstää. Yleisesti ylläpitosopimukset jaetaan kahteen ryhmään riippuen siitä, onko kyseessä valmisohjelmisto vai asiakaskohtainen ohjelmisto.

Valmisohjelmiston ylläpitoa tarjotaan samoilla sisältö- ja hinnoitteluperusteilla kaikille valmisohjelmiston käyttäjille ja asiakas voi yleensä vaikuttaa ylläpitoon vain valitsemalla määritellyistä palvelutasoista tälle sopivimman vaihtoehdon. Vastuu kehitystyöstä valmisohjelmistojen kohdalla on yksin toimittajalla, joka julkaisee ohjelmistosta uusia päivitysversioita sitä mukaa, kun ohjelmiston virheitä on korjattu ja uusia toimintoja on ohjelmistoon lisätty.

Asiakaskohtaisen ohjelmiston kehittämis- ja virheidenkorjausvastuu on asiakkaalla, jolle toimittaja varaa ylläpitosopimuksen puitteissa tietyn resurssimäärän. Tätä resurssia asiakas voi käyttää kehittääkseen ohjelmistoa ja poistaakseen ohjelmistosta havaitsemiaan virheitä ja toimittaja laskuttaa asiakasta ylläpidosta tehtyjen tuntien mukaan.

### 2.4.3 Tukisopimus

Tukisopimus sisällytetään yleensä ohjelmiston ylläpitosopimukseen, mutta se voidaan tehdä myös erillisenä sopimuksena. Sopimus määrittää, kuinka käyttöä toimittajan puolesta opastetaan ja kuinka esiintyneitä ongelmatilanteita selvitetään asiakkaan apuna. Ideana yleensä on, että asiakkaalle on määritetty jokin kontaktikanava toimittajan organisaatioon, jonka välityksellä pyritään ratkaisemaan asiakkaalla ilmenevä ongelma. Jos ongelmaa ei pystytä välittömästi ratkaisemaan, niin toimittaja on yleensä velvoitettu selvittämään ongelman alkuperä ja informoimaan tästä asiakasta.

## 2.5 Palvelusopimukset

Nykypäivänä yritykset pyrkivät optimoimaan toimintaansa siten, että he keskittyvät ainoastaan ydinliiketoimintaansa ja hankkivat muut tukitoiminnat

palveluina ulkopuoliselta palveluntarjoajalta.

Palvelun hankkimisessa ulkopuoliselta palveluntarjoajalta on olemassa kahta eri toteutustapaa, jotka ovat käyttöpalvelu ja sovellusvuokraus. Pekka Takki selventää käyttöpalvelukäsitettä kertoen sen olevan järjestely, jonka perusteella ulkopuolinen tietotekniikkatoimittaja ottaa huolehtiakseen asiakkaan tietojenkäsittelyyn liittyvien, sopimuksessa määriteltyjen toimintojen ja järjestelmien ylläpitämisestä tai tarjoaa tietojenkäsittelypalveluita joko osittain tai kokonaan omia resurssejaan hyödyntäen [16]. Sovellusvuokrauksessa asiakas ostaa ohjelmiston toimittajalta oikeuden käyttää ohjelmistoa sopimuskauden ajan. Molempien toteutustapojen kohdalla tulee sopimusneuvotteluissa kiinnittää erityistä huomiota tietoturvan toteutukseen ja erilaisten salaisten tietojen, kuten henkilötietojen, välittämiseen, sillä asiakasyritys antaa sopimuksen myötä palveluntarjoajalle pääsyn asiakasyrityksen kriittisiin tietoihin.

### 2.5.1 Käyttöpalvelu

Yrityksen ulkoistaessa järjestelmiensä hallintaa ja ylläpitoa ulkopuoliselle palveluntarjoajalle solmitaan yrityksen ja palveluntarjoajan välille käyttöpalvelusopimus. Tämä sopimus pitää sisällään kuvauksen niistä toiminnoista, jotka siirtyvät sopimuksen astuessa voimaan asiakasyrityksen hallinnasta palvelua tarjoavan toimittajan ylläpidettäviksi, palvelutason mittaustavoista, sopimuksen voimassaoloajan ja yleensä palvelutasoliitteen, joka määrittää konkreettiset raja-arvot palvelulle ja niiden noudattamattomuuksista aiheutuvat sanktiot palvelun toimittajalle.

### 2.5.2 Sovellusvuokraus

Sovellusvuokrauksessa asiakas ostaa ohjelmiston tuottamaa palvelua, eikä ohjelmistoa itsessään. Sopimuksen kohteena onkin näin ollen perinteisistä ohjelmistosopimuksista poiketen ohjelmiston tuottama palvelu. Sovellusvuokrauksessa asiakas ei tarvitse erillistä ylläpitosopimusta, sillä sovellusta vuokraava toimittaja, palveluntarjoaja, huolehtii ohjelmiston ylläpidosta ja asiakkaan laiteinvestointitarpeet vähenevät tästä johtuen huomattavasti. Sovellusvuokrausmallin yksi suuri etu aikaisempiin ohjelmistojen toimitusmalleihin verrattuna on Pekka Takin mukaan se, että myös pienyrityksillä on mahdollisuus ottaa käyttöön ennen vain suuryrityksien saatavilla olleita ohjelmistokokonaisuuksia kohtuulliseen hintaan [16].

### 2.5.3 Palvelutasoliite

Asiakas ja ohjelmiston toimittaja määrittelevät ohjelmiston käytettävyydelle käyttösopimuksen palvelutasoliitteessä reunaehdot, joiden sisällä ohjelmiston tulee toimia sopimuksen voimassaolon ajan. Palvelutasoa mitataan yhteisesti määritetyin parametrein, kuten ohjelmiston käytön nopeudella, virhetilanteiden määrällä ja ohjelmiston saatavuudella. Jos havaitaan, että ohjelmisto ei

ole täyttänyt mitatulla ajanjaksolla sovittua palvelutasoa, on ohjelmiston toimittaja velvoitettu tapauksesta riippuen alentamaan palvelustaan perimäänsä maksua tai maksamaan asiakkaalle sopimuksen mukainen sopimusrikkomussakko. Tämä menettely turvaa asiakkaalle toimivamman palvelun ja motivoi ohjelmiston toimittajaa panostamaan ohjelmiston aktiiviseen tilanseurantaan ja virheiden korjaukseen.

## 2.6 Asiakaspalvelu

Tietojärjestelmän käyttäjälle tulee väistämättä eteen tilanteita, joista hän ei selviä ilman apua eteenpäin. Tällaisia tilanteita varten tietojärjestelmälle pyritään jo kehitysvaiheessa laatimaan käyttöohje ja ratkaisumalleja oletettuihin ongelmatilanteisiin, mutta kaikkiin tilanteisiin ei ole olemassa yksiselitteistä ratkaisua ja käyttöohje voi osoittautua riittämättömäksi käyttäjälle. Tämä ongelma korostuu myös siitä syystä, että tietojärjestelmä koostuu useasta erillisestä osa-alueesta, jotka ovat vuorovaikutuksessa toisiensa kanssa ja näillä on hyvin todennäköisesti eri toimittaja ja näinollen erilaiset käyttöohjeet ja toimintamallit. Tietojärjestelmän käyttöönoton vaiheisiin, ohjeistuksiin ja muihin toimintoihin paneutuu syvällisemmin Mattila lopputyössään [12].

Toimittajan ja asiakkaan välisestä ylläpito- tai käyttösopimuksesta riippuen käyttäjän ensikontakti ongelmatilanteessa on joko toimittajan kouluttama asiakkaan yhteyshenkilö tai toimittajan oma asiakaspalveluhenkilö.

Käyttäjän ongelmatilanne pyritään selvittämään aina ensikontaktin aikana, mutta joissain tapauksissa ongelma vaatii syvempää perehtymistä ja tämä ongelma saattaa osoittautua virheeksi ohjelmistossa, laitealustassa, tiedonsiirtoyhteyksissä tai jossain muussa tietojärjestelmän osassa, mikä tarvitsee korjausta. Asiakkaalle ylläpitosopimuksessa sovittu asiakaspalvelu selkiyttää toimintaa ongelmatilanteissa ja antaa turvaa jatkuvuudesta, sillä ongelmista päästään ylitse joko neuvonnalla, parannetuin ohjeistuksin tai virheet korjaamalla.

Toimittajan näkökulmasta katsottuna käytön opastaminen ja ohjeistaminen antaa tietojärjestelmästä paremman imagon, parantaa toimittajan julkisuuskuvaa ja ylläpitää hyviä asiakassuhteita.

## 2.7 Korjaava ylläpito

Korjaavaan ylläpitoon luokitellaan sellaiset ylläpitotehtävät, joiden tarkoituksena on korjata löytyneitä virheellisiä toimintoja ja puutteita toiminnoissa.

Ylläpitosopimukseen määritetään havaituille virheille luokitus, jonka mukaan esimerkiksi kriittinen virhe korjataan ensi tilassa, kun taas muutosehdotukseksi luokiteltava virhe vasta sopivan ajankohdan tullen. Seuraava taulukko olkoon esimerkkinä tällaisesta virheiden luokittelusta.

Taulukko 2.1: Virheiden määrittely

| Virheluokitus | Määrittely |
|---|---|
| 1-kriittinen | Tuotannon estävä |
| 2-vakava | Tuotantoa vakavasti haittaava |
| 3-haittaava | Määrittelyn vastainen - ohitettavissa |
| 4-lievä | Kosmeettinen virhe |
| 5-muutosehdotus | Muutos, parannus |

Samainen ylläpitosopimus voi myös määrittää tietojärjestelmälle takuuajan, jonka puitteissa havaitut virheet korjataan veloituksetta toimittajan toimesta, mutta jonka jälkeen kaikenlainen muokkaaminen on erikseen asiakkaalta laskutettavaa.

Virheen havaitsemisen jälkeen päätetään, voidaanko virhe ohittaa pienellä muutoksella, saadaanko virhe kierrettyä käyttäjien ohjeistuksella tilapäisesti vai tarvitaanko tietojärjestelmästä uusi kehitysversio virheen korjaamiseksi. Tämän jälkeen ohjeistusta, tietojärjestelmän sisältöä tai molempia muuttamalla aikaansaatu uusi versio luovutetaan asiakkaan käyttöön.

## 2.8 Mukauttava ylläpito

Mukauttavan ylläpidon tarkoituksena on taata tietojärjestelmän toiminta muuttuvassa ympäristössä. Toimittaja pyrkii optimoimaan ohjelmiston yhteensopivuutta uuteen tietojärjestelmän laitteistokokoonpanoon tai vaikkapa käyttöjärjestelmään. Yleensä tällainen kehitystoimista voidaan sisällyttää ylläpitokustannuksiin ja näin ohjelmiston oleelliset päivitykset välittyvät myös asiakkaalla olevaan versioon. Mukauttava ylläpito ei vaikuta tietojärjestelmän alkuperäisiin toiminnallisuuksiin. Kun ympäristön tuomia muutospaineita ei voida enää kohtuullisella muutostyöllä tuottaa tietojärjestelmän ohjelmistoon, on ohjelmisto elinkaarensa päässä ja se on korvattava uudella ohjelmistolla, joka kykenee suoriutumaan ympäristön sille asettamista vaatimuksista.

## 2.9 Kehittävä ylläpito

Tarve ominaisuuksien lisäämiseen valmiiseen tietojärjestelmään voi olla lähtöisin joko toimittajalta tai asiakkaalta ja tästä erosta johtuen puhutaan joko laajentamisesta tai muutoksesta.

Laajentamisessa toimittaja toteuttaa uusia ominaisuuksia tietojärjestelmän ohjelmistoon tai laitteistoon ja riippuen asiakkaiden ylläpitosopimuksien versiointisäännöistä asiakkaat joko uusivat tietojärjestelmän uusimpaan versioon tai pitäytyvät vanhemmassa versiossa, jolloin asiakkaan kustannuksia säästyy niin uudelleenkoulutuksen kuin versionvaihdon kustannuksien avulla.

Muutettaessa tietojärjestelmää on asiakkaalle syntynyt tarpeita, joiden täyttämiseen nykyinen tietojärjestelmä ei kykene. Tällaiset muutostyöt toteute-

taan pääosin erillisenä projektina, mutta joissain tapauksissa ylläpitosopimus voi sisältää pienen määrän asiakkaan esittämistä muutosehdotuksista.

## 2.10 Ennaltaehkäisevä ylläpito

Toimia, joiden avulla pyritään parantamaan tietojärjestelmän ylläpidettävyyttä ja takaamaan sen toimintaa erinäisissä tilanteissa, luokitellaan kuuluviksi ennalta ehkäisevään ylläpitoon. Tällaisia toimia ovat valvonnan edistäminen ja dokumentaation parantaminen, jotta tulevaisuudessa voidaan havaittuja ongelmia korjata nopeammin tai jopa ehkäistä niiden syntymiset kokonaan. Tietojärjestelmästä hankitaan tietoutta sen käytön myötä ja havaitut erityispiirteet kirjataan ylös yleisien käyttäytymismallien selvittämiseksi.

Kerätyn informaation avulla voidaan reagoida mahdollisiin havaittavissa oleviin ongelmatilanteisiin jo ennen niiden varsinaista ilmentymistä ja näinollen vähentää ongelman tietojärjestelmälle aiheuttamaa haittavaikutusta.

Myös käyttäjien usein toistuviin ylläpidollisiin kysymyksiin pyritään koostamaan yleispäteviä ratkaisumalleja, jolloin näiden ohjeiden julkaiseminen vähentää ohjelmiston ylläpidollista kuormitusta.

## 2.11 Yhteenveto

Tietojärjestelmä luo suoritettavalle ohjelmistolle ympäristön tämän toimintaa varten. Tietojärjestelmä jaetaan yleensä tiedonkäsittelylaitteistoon, tiedonsiirtolaitteistoon ja ohjelmistoalustaan. Ylläpidon sisältö vastuukysymyksineen ja erilaisien toimintamallien osalta määritellään asiakkaan kanssa useiden sopimusten avulla. Ylläpitosopimuksiin sisältyvät ylläpidon toteutuksen määrittelyt ja palvelusopimuksiin kyseessä olevan ohjelmiston tuottaman palvelun yleiset laatukriteerit ja standardit. Tietojärjestelmän ylläpito voi sisältää korjaus-, muutos- , kunnossapito- ja kehitystöitä.

# Luku 3

# Ennaltaehkäisevä ylläpito

Ohjelmisto- ja laitealustasta muodostuva tietojärjestelmäkokonaisuus vaatii huolellista suunnittelua ja aktiivista seurantaa vakaan ja toimivan kokonaisuuden aikaansaamiseksi. Ehkäisevä ylläpito voidaan jakaa valvontaan, ohjeistuksen parantamiseen ja ongelmatilanteisiin varautumiseen. Uuden järjestelmän ylläpito on pääasiassa näinollen huoltotoimintaa, mutta ajan myötä muutosten ja kehittämisen tarve suurenee, kuten Kalliala painottaa tutkielmassaan [8].

Tietojärjestelmän ohjelmisto tarvitsee toimiakseen infrastruktuurin, joka sisältää laitealustan, ohjelmistoalustan ja tietoliikenneyhteydet. Laitealusta, jolla ohjelmistoa ajetaan, voi olla hyvin erilaisessa roolissa, jos ohjelmisto on toteutettu laitteen hallintaa varten tai jos ohjelmisto on suunniteltu esimerkiksi tiedonhallintaan, sillä laitetta hallitseva ohjelmisto käyttää laitetta toteuttaakseen tehtäväänsä, kun taas jälkimmäisessä tapauksessa laitealusta antaa mahdollisuuden suorittaa ohjelmistoa. Ohjelmistoalustalla tarkoitetaan ohjelmistoa, joka antaa halutulle ohjelmistolle puitteet toimintaa varten. Jos ohjelmisto on tehty laitteen hallintaan, ei sillä yleensä ole enää erillistä ohjelmistoalustaa. Ohjelmistoalustana voidaan pitää käyttöjärjestelmää ja vielä spesifisemmästä ohjelmistoalustasta esimerkkinä olkoon sovelluspalvelimet, jotka suorittavat toteutettua ohjelmistoa suoden sille rajapinnat verkkokommunikointiin ja levyhallintaan.

Tietojärjestelmässä esiintyvät ongelmat heijastuvat tiedon käsittelyyn joko häiriten sitä tai estäen sen kokonaan. Mahdollisten epäselvyyksien välttämiseksi tietojärjestelmän käyttämisessä on toimittajan määriteltävä tarkasti asiakkaan kanssa mitkä järjestelmän osiot kuuluvat ylläpidoltaan asiakkaan ja mitkä toimittajan vastuualueeseen. Yleisesti ottaen voidaan tietojärjestelmän ylläpidosta pitää vastuullisena sitä tahoa, joka suorittaa tiedon käsittelyä eli jos asiakas on hankkinut kokonaisuuden palveluna, vastuu tietojärjestelmän ylläpidosta on toimittajalla ja muutoin asiakkaalla.

Laitealustan ehkäisevä ylläpito itsessään käsittää fyysisen laitteiston valvonnan, sen huoltamisen, päivittämisen ja erilaisten yhteyksien valvonnan ja kunnostamisen. Laitealustassa ilmenevä ongelmatilanne aiheuttaa mitä to-

dennäköisimmin ongelmia ohjelmiston suorittamiseen estäen sen pahimmassa tapauksessa kokonaan.

Ohjelmistoalustan ylläpito tarkoittaa sen muistin- ja levytilan käytön ja virhetilanteiden seurantaa. Ohjelmistoalustassa tapahtuva ongelmatilanne ei välttämättä vaikuta suoritettavaan tiedonkäsittelyyn millään tavalla, mutta myös ohjelmistoalustan ongelmatilanne, kuten muistin loppuminen, estää ohjelmiston toiminnan kokonaan kuten laitealustassakin tapahtuva vakavampi ongelma.

Esilletulevien tarpeiden toteuttaminen ja uusien kehittyneempien ominaisuuksien lisääminen vanhaan järjestelmään asettaa muuttuvan ympäristön myötä ylläpidollisesti haastavan tehtävän, sillä päivittäminen tulee toteuttaa aiempaa toiminnallisuutta rikkomatta ja ylläpidettävyyden tulee säilyä päivityksen jälkeenkin. Muutosten dokumentointi ja testaaminen ovatkin päivityksien yhteydessä tärkeässä asemassa, jotta toimivuus voidaan taata myös muutostöiden jälkeen.

Tämän luvun tarkoituksena on tutustuttaa tietojärjestelmän ehkäisevään ylläpitoon, sillä tietojärjestelmässä ilmenevät ongelmat heijastuvat välittömästi siinä ajettuihin ohjelmistoihin ja tästä johtuen tietojärjestelmän ylläpidolla on suuri merkitys toimivan kokonaisuuden aikaansaamiseksi ja ylläpitämiseksi. Tietojärjestelmää, joka tarjoaa ohjelmistolle infrastruktuurin toimimista varten, on pystyttävä valvomaan ja siihen kohdistuvia ongelmatilanteita on pystyttävä ehkäisemään ja korjaamaan siten, että se takaa ohjelmistolle puitteet toimia sille asetettujen ehtojen mukaisesti.

## 3.1 Saatavuuden takaaminen

Saatavuus tietojärjestelmien kohdalla tarkoittaa, että miten suuren osan kokonaisajasta järjestelmä pystyy suoriutumaan sille annetusta tehtävästä vallitsevissa olosuhteissa. Saatavuus on näinollen laskettavissa toimivuusajan ja kokonaisajan suhteena, jossa kokonaisaika on järjestelmän toimivuusajan ja toimimattoman ajanjakson summa. Tämä määritelmä voidaan myös esittää kaavana 1.

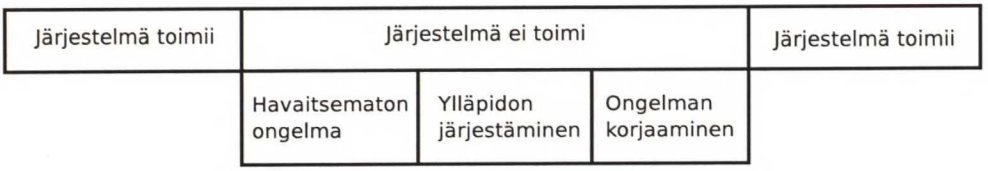
**Algorithm 1** Saatavuuden määritelmä

$$Saatavuus \equiv \frac{Toimivuusaika}{Kokonaisaika} \cdot 100\,\%$$

$$Kokonaisaika \equiv Toimivuusaika + Toimimaton\,aika$$

$$Saatavuus \equiv \frac{Toimivuusaika}{Toimivuusaika + Toimimaton\,aika} \cdot 100\,\%$$

Toimimaton ajanjakso kahden toimivan ajanjakson välillä voidaan jaotella aikaan, jolloin järjestelmässä on tapahtunut virhe, mutta sitä ei olla havaittu, ylläpidon järjestämiseen ja ongelman korjaamiseen. Tätä havainnollistaa kaavio 3.1, jonka Kaikkonen esittelee lopputyössään [7].




Kuva 3.1: Toimimattoman ajanjakson muodostuminen [7]

Saatavuus on yhtä tärkeä osatekijä kuin järjestelmän tekninen suorituskykykin mitattaessa järjestelmän kokonaissuorituskykyä, toteaa Kaikkonen lopputyössään [7]. Saatavuus voidaan jakaa järjestelmän luotettavuuteen, ylläpidettävyyteen ja huollettavuuteen.

Tietojärjestelmissä esiintyy vikoja epäsäännöllisin väliajoin, jotka aiheutuvat joko ohjelmallisista tai laitealustaongelmista. Toimittajan ja asiakkaan väliset sopimukset tietojärjestelmän ylläpidosta ja palvelun tasosta ajavat yhdessä kohti tarkemmin valvottua ja vikasietoisempaa järjestelyä. Asiakas maksaa ylläpidosta, jotta järjestelmä olisi aina asiakkaan käytettävissä eli saatavissa ja toimittaja pyrkii aktiivisesti ehkäisemään ja minimoimaan ongelmia, sillä etenkin asiakasta haittaavista ongelmista toimittaja on velvollinen antamaan käyttömaksualennuksia tai peräti sopimusrikesakkoja, joskin näissä on sopimuskohtaisia eroja.

Tietojärjestelmän hajauttamisen ja varajärjestelmien merkitys korostuu, jos hallinnoitavana olevan järjestelmän on täytettävä tiukat saatavuuskriteerit, joista hyvänä esimerkkinä olkoon lennonjohdon käytössä olevat järjestelmät. Tällaisissa tilanteissa on tärkeää varmistaa, että vaikka jossain laitealustan osassa tai ohjelmistossa ilmenisikin virhetilanne, näiden aiemmin suorittamat toiminnot voidaan siirtää jollekin muulle järjestelmän osalle siten, ettei loppukäyttäjä välttämättä edes huomaa tapahtunutta siirtoa. Tietojärjestelmän rakennetta suunniteltaessa on mietittävä halutun saatavuustason edellyttämän varajärjestelmän suhdetta sen mukanaan tuomiin kustannuksiin. Kuten Christian kumppaneineenkin huomauttaa, ydinkysymys on löytää tasapaino virheenhavainnoinnin, palautumisen, ylimääräisen varakapasiteetin ja näistä koituvan hyödyn suhteen [3].

### 3.1.1 Ongelmatilanteiden välttäminen

Ohjelmiston ja tietojärjestelmän toimintaa ennakoimalla pyritään ehkäisemään mahdollisten ongelmien syntymisiä taaten täten asiakkaalle laadukkaampaa palvelua ja toimivampaa järjestelmää. Laitealustan osalta tämä ennakointi tarkoittaa varajärjestelmien toteuttamista, laitealustan kriittisien komponenttien kahdentamista, laitealustan säännöllistä huoltamista ja komponenttien uusimista, jotka parantavat laitteiston toimintavarmuutta. Ohjelmistoalustan osalta ongelmatilanteiden ennakointi tarkoittaa mahdollisien ongelmatilanteiden identifioimista ja niiden ehkäisemistä ennen niiden vaikuttamista toiminnassa oleviin ohjelmistoihin.

Maksaessaan ylläpidosta asiakas todellisuudessa rahoittaa tätä ylläpidol-

lista ennakointia, sillä itse ongelmatilanteet ovat ajallisesti lyhytkestoisia ja niiden esiintymistiheys on vähäinen ohjelmiston käyttöikään verrattuna. Huolellisella ennakoinnilla näitä ongelmatilanteita kyetään kuitenkin vielä vähentämään.

### 3.1.2 Ongelmatilanteisiin varautuminen

Vaikka ongelmatilanteita pyritään ehkäisemään, on jatkuvuuden kannalta hyväksi varautua sellaisten varalle. Tämä ongelmatilanteisiin varautumisen suunnitelma tunnetaan varautumissuunnitelmana.

Varautumissuunnitelma tarkentaa millaisia uhkia tietojärjestelmän toiminnan estymiselle on olemassa, kuinka niitä voidaan havaita ja miten niistä kyetään palautumaan takaisin toimivaan tietojärjestelmään. Tarkalla ongelmien määrittämisellä voidaan lyhentää järjestelmän toimimattoman ajanjakson 3.1 kahta ensimmäistä osaa, sillä näiden ongelmien havaitseminen nopeutuu, jos niitä osataan odottaa ja myöskin ylläpitotoimiin voidaan ryhtyä aiempaa nopeammin jos ongelmatilanteen selvittämiseksi on olemassa valmiit toimintamallit.

### 3.1.3 Ongelmatilanteiden selvittäminen

Laitealustan ongelmat esiintyvät yleensä ennaltavaroittamatta etenkin arvioidun käyttöiän loppupuolella eikä näitä koskaan pystytä täysin estämään. Laitealustan ongelmat heijastuvat koko järjestelmään, kuten esimerkiksi prosessorituulettimen rikkoutuessa prosessori kytkeytyy vikatilaan aiheuttaen koko järjestelmän toimintojen näennäisen pysähtymisen, jolloin ongelman selvittämiseksi on toiminnan oltava toimittajan kannalta koordinoitua, hyvin ohjeistettua ja jouhevaa. Näin toimittaessa järjestelmä saadaan palautettua toimintakuntoon lyhyemmässä ajassa ja estetään ongelman leviäminen laajemmalle toimiviin laitealustan osakokonaisuuksiin.

Ohjelmistoalustan ongelmat kehittyvät pidemmällä aikavälillä laitealustan ongelmiin nähden ja näiden syntyä voidaan havainnoida aktiivisella valvonnalla, kuten massamuistin käyttöasteen tai prosessorikuormituksen kehityksen seurannalla. Ongelma voidaan yksiselitteisesti havaita vertaamalla nykyistä tilannetta aiempaan tilanteeseen, mutta ongelman selvittäminen ohjelmistoalustassa on paljon monimutkaisempaa laitealustan ongelmiin verrattaessa. Ohjelmistoalustan ongelman selvittämiseksi on rajattava ongelma-aluetta tutkimalla laite- ja ohjelmistoalustan välisen rajapinnan toimintaa, ohjelmistoalustan ja sen ohjelmistojen välisten rajapintojen toimintaa ja näiden lisäksi eri ohjelmistojen välisiä mahdollisia yhteensopivuusongelmia. Ongelman tarkkaa kohdistusta ei välttämättä pystytä koskaan tekemään ja tämä hankaloittaa tilanteen normalisointia.

Ylläpidollisesti laitealustan viat ovat äkillisiä ja niiden selvittäminen on suoraviivaista, kun taas ohjelmistoalustan viat havaitaan pidemmän aikajakson aikana ja niiden selvittäminen on aikaa vievää ja ongelman ytimen rajaa-

minen haasteellista.

## 3.2 Palvelun kehittäminen

Palvelun kehittäminen tietojärjestelmätasolla tarkoittaa tehokkuuden ja toimintavarmuuden parantamista. Kehittämällä voidaan kasvattaa järjestelmän kykyä sietää suurempia käyttäjämääriä, alentaa yksittäisen käyttäjän kokemia suoritusaikoja ja varmentaa palvelujen luotettavuutta. Palvelua kehitetään laitetasolla esimerkiksi lisäämällä prosessointitehoa, muistikapasiteetin määrää ja nostamalla verkkoyhteyden nopeuksia. Ohjelmistoalustan kehittämisessä toimintojen toteutusta tehostetaan optimoimalla ohjelmiston rakennetta ja tukemalla laitealustan ominaisuuksia aikaisempaa tehokkaammin. Yleisesti näitä ominaisuuksia kuitenkin kehitetään vasta todellisen tarpeen niin vaatiessa, sillä nämä aiheuttavat kehityshankkeet menoeriä järjestelmän toimittajalle.

White kumppaneineen suosittavat yhdenmukaistamaan ohjelmisto- ja laitealustaa, kehittämään työkaluja ongelmien selvittämistä varten, parantamaan häiriötilanteiden havainnointia ja raportointia sekä keskittämään kehitystyötä havaituista ongelmista vakavimpiin [22]. Tämä Whiten ja kumppanien kehityssuuntaussuositus tähtää kokonaisuuteen, jossa ongelmatilanteet havaitaan mahdollisimman pian tämän esiintymisen jälkeen ja ongelman aiheuttaja pystytään identifioimaan ja mahdolliset ongelmanratkaisumallit on jo kehitettyinä ennakkoon. Tämän lisäksi tietojärjestelmän komponenttien homogenisoinnilla voidaan vähentää mahdollisia ongelmakohtia ja tietojärjestelmän tietämystä voidaan syventää enemmän verrattuna monista erilaisista laite- ja ohjelmistoratkaisuista koottuihin tietojärjestelmiin verrattuna.

## 3.3 Laitealustan valvonta

Laitealustan toiminnassa esiintyvät ongelmatilanteet vaikuttavat aina suoraan siinä ajettavien ohjelmistojen toimintaan estäen niiden käytön kokonaan tai osittain. Laitealustan ongelmat ovat joko kulumisesta, laitteiston käytöstä tai ulkopuolisesta häiriöstä johtuvia ja niiden korjaaminen on yleensä hidas operaatio.

Laitealustaan kuuluvat osat ovat kuluvia ja niiden elinikä on rajallinen. Laitealustaan tulee vikoja ajan myötä eikä tältä voi välttyä, joskin laitteistovikoja voi minimoida uusimalla laitteistoa säännöllisin väliajoin ja monistamalla laitealustan kriittisiä komponentteja vahingoittumisriskin pienentämiseksi.

Yksittäisen laitealustan komponentin toimivuuden todennäköisyys ei riipu toisien komponenttien tilasta. Toimimattomuuden ja toimivuuden riippuvuus voidaan esittää todennäköisyyslaskennan opein kaavan 2 mukaisesti. Jos laitealustan osat ovat toisistaan riippumattomia, voidaan kaavan 3 tuloksista havaita, että jos laitealustassa oleva komponentti on mahdollista monistaa, todennäköisyys tämän kokonaisuuden toimivuudelle kasvaa. Tästä olisi mah-

dollista tulkita, että monistamalla komponentteja äärettömän monta kertaa, saavutettaisiin vikaantumaton laitealusta. Laitteistoalustan kustannukset rajoittavat kuitenkin toteutettavan laitteiston kokoonpanoa ja toiseksi äärettömän kokoisen laitealustan hallinnointi ja ylläpito muodostuisivat ylitsepääsemättömiksi ongelmiksi.

**Algorithm 2** Toimivuuden ja toimimattomuuden keskinäinen todennäköisyyssuhde

$P(\Omega)$=1, varma tapaus

$0 \leq P(toimivuus) \leq P(\Omega)$

$P(toimivuus) + P(toimimattomuus) = P(\Omega)$

$P(toimivuus) = P(\Omega) - P(toimimattomuus)$

**Algorithm 3** Toisistaan riippumattomien komponenttien vikaantumistodennäköisyys ajan hetkellä t

Komponentin A vikaantumistodennäköisyys ajan hetkellä t: $P(A)$
Komponentin B vikaantumistodennäköisyys ajan hetkellä t: $P(B)$

Yhteinen vikaantumistodennäköisyys ajan hetkellä t: $P(A \cap B) = P(A)P(B)$

Käytännön esimerkki:
Komponentin A vikaantumistodennäköisyys ajan hetkellä t: $P(A) = 0,005$
Komponentin B vikaantumistodennäköisyys ajan hetkellä t: $P(B) = 0,005$

Yhteinen vikaantumistodennäköisyys ajan hetkellä t:
$P(A \cap B) = P(A)P(B) = 0,005 * 0,005 = 2,5 * 10^{-5}$

Laitteiston käyttö voi ajaa laitealustan tilanteeseen, joka voi pysäyttää sen toiminnan kokonaan, vaikka laitteistossa ei varsinaista vikaa olekaan. Tällaisia tilanteita syntyy, jos laitealustaa rasitetaan jatkuvasti, eikä laitteiston tilaa seurata riittävän tehokkaasti. Laitealustan käyttömuisti, massamuisti tai prosessointikyky saattavat loppua kesken ja pahimmillaan tilanne kehittyy hallitsemattomaksi eikä laitealustalla voida suorittaa mitään ennen tilanteen täydellistä nollaamista.

Laitealusta reagoi myös ulkopuoliselle häiriölle, kuten sähkökatkoksiin tai tiedonsiirtoyhteyksien ongelmiin. Sähkökatkokset pysäyttävät koko laitealustan toiminnan, jollei tähän olla varauduttu apuvirtalähteiden avulla. Tiedonsiirtoyhteyksien ongelmat puolestaan pysäyttävät useimpien ohjelmistojen toiminnan eikä tälle välttämättä ole löydettävissä ratkaisua muutoin kuin asentamalla täysin toisistaan riippumattomat tiedonsiirtoyhteydet laitealustaan tai monistamalla laitealustaa siten, että edes osa laitteistokokonaisuudessa sijaitsevat eri tiedonsiirtoyhteyksien äärellä.

Laitealustan valvonta osoittautuu hyvin laaja-alaiseksi kapeallakin tarkastelulla ja sen ylläpitäminen on haasteellista, sillä yleensä ongelmat esiintyvät varoittamatta ja ongelmien tarkka paikantaminen voi tuottaa ongelmia.

Laitealustan ylläpitämiseksi on huolehdittava ympärivuorokautisesta valmiustilasta, jotta ilmeneviin ongelmiin pystytään reagoimaan mahdollisimman nopeasti ja tehokkaasti. Laitealustaan voidaan kiinnittää erilaisia antureita ja mittareita, joiden avulla voidaan tuottaa arvioita mahdollisen ongelmatilanteen todennäköisyydestä, mutta nämä eivät koskaan pysty täydelliseen ennustukseen.

## 3.4 Ohjelmistoalustan valvonta

Ohjelmistoalusta toimii tietojen käsittelyyn toteutetun ohjelmiston rajapintana laitealustan suomille toiminnoille ja on itse ohjelmiston tavoin mahdollinen ongelman aiheuttaja toimintaketjussa. Koska yleisesti ohjelmistoalustalle annetaan rajoitetut oikeudet laitealustan käyttöön, voi ohjelmistoalustassa ilmetä samankaltaisia oireita kuin itse laitealustassakin eli ohjelmistoille ei välttämättä riitä käyttömuistia tai riittävästi prosessointikapasiteettia, vaikka itse laitealusta toimiikin vielä moitteetta. Ohjelmistoalustan ja suoritettavan ohjelmiston yhteensopivuus voi myös tuottaa ongelmatilanteita, jos ohjelmistossa yritetään käyttää ohjelmistoalustan ominaisuuksia virheellisesti tai jos ohjelmistoalusta reagoi tiettyihin syötteisiin spesifikaatioista poiketen.

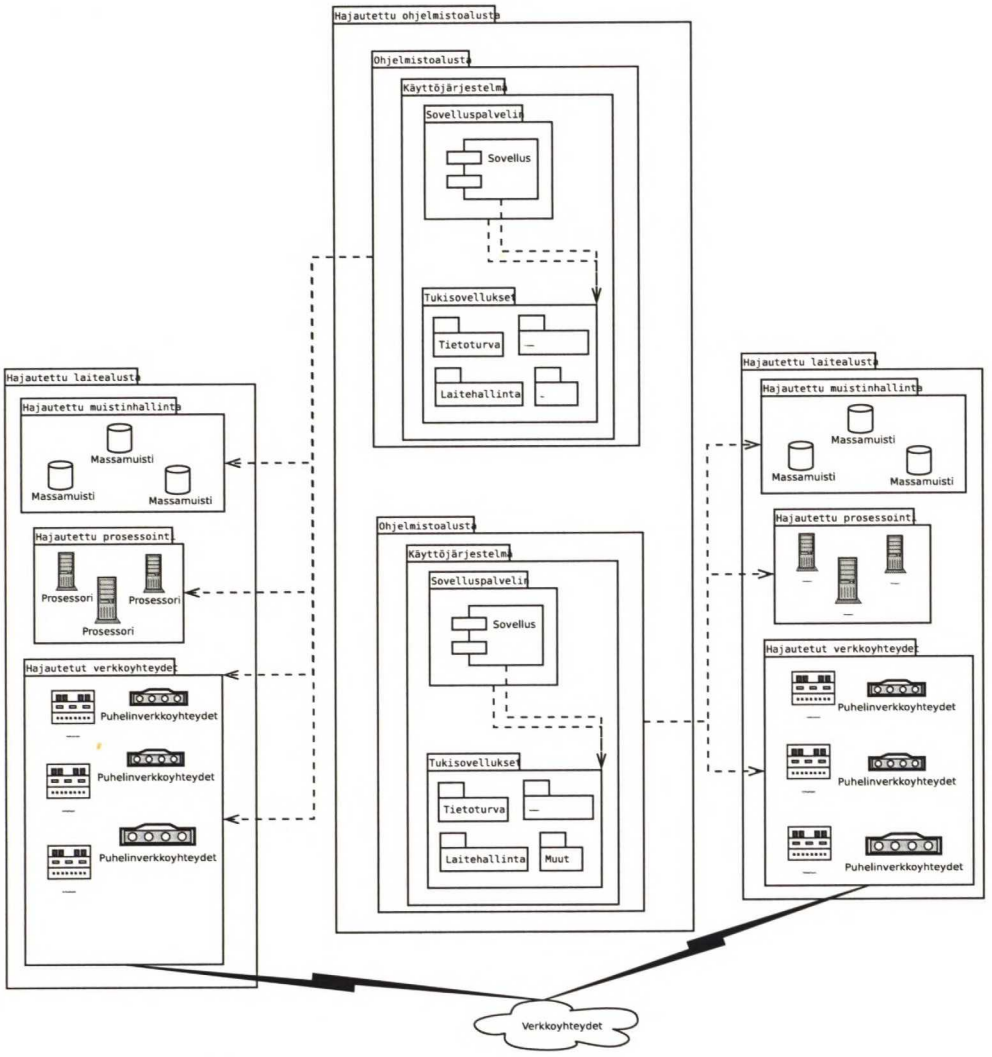
Ohjelmistoalustan valvonta on hyvin tapauskohtaista ja tietojen kerääminen ongelmatilanteen havaitsemiseksi on täysin ohjelmistonalustatoteutuksesta riippuvainen. Yleisesti ongelmatilanteen selvitystyö lähtee suoritettavan ohjelmiston toimintojen tulkitsemisesta ja jos tämän toiminnassa ei ole tavallisuudesta poikkeavaa, siirrytään tutkimaan ohjelmistoalustan toimintaa eli ohjelmistoalustaa itsessään ei valvota aktiivisesti.

## 3.5 Hajautetun ohjelmisto- ja laitealustan valvonta

Ohjelmisto- ja laitealustan hajauttamisella pyritään parantamaan ohjelmiston saatavuutta eli ohjelmisto kykenee palvelemaan asiakkaan tarpeita pidempiä ajanjaksoja yhtäjaksoisesti ilman katkoksia toiminnoissa. Kun laitealusta tai ohjelmistoalusta tai molemmat yhdessä on toteutettu hajautetusti, pienenee todennäköisyys ohjelmiston toimivuutta haittaavien ongelmien esiintymiselle, sillä ohjelmiston suorittaminen voi jatkua hajautetun järjestelmän toimivissa osissa myös ongelmatilanteiden esiintyessä jossain osassa järjestelmää. Koska hajautetussa järjestelmässä kukin hajautettu osio on toisistaan riippumaton, voidaan tässäkin tilanteessa hajautetun järjestelmän käyttämisen yhtenä perusteluna käyttää kaavan 3 antamaa tulosta.

Vaikka hajauttamisella saavutetaan toimintavarmempi ympäristö asiakkaan näkökulmasta, on sen ylläpitäminen paljon haasteellisempaa, koska missä tahansa järjestelmän osassa saattaa piillä ongelmia, jotka hajautuksen vuoksi eivät näy ylläpidolle. Ylläpidon on aktiivisesti seurattava jokaisen laite- ja

ohjelmistoalustan toimintaa yhtaikaisesti, jolloin kokonaiskuvan hahmottaminen on hankalampaa. Hajautetusta tietojärjestelmästä on havainnekuvana kuva 3.2.




Kuva 3.2: Havainnekuva hajautetusta tietojärjestelmästä

Laitealustan hajauttamisessa alustan komponentteja tai osakokonaisuuksia monistetaan siten, että käytettävien ohjelmistojen näkökulmasta se näyttää yhdeltä laitealustalta ja näiden keskinäinen toimintojen jakaminen on toteutettu laitealustan ytimen logiikalla. Häiriöiden tai ongelmien havaitseminen laitealustakonfiguraatiossa vaatiikin näinollen tarkkaavaisuutta, sillä vaikka järjestelmä toimii moitteetta, voi esimerkiksi osa massamuistilaitteista olla epäkunnossa.

Ohjelmistoalustan hajauttamisessa hajautus tehdään ohjelmallisesti eli ohjelmistoalustaa suoritetaan useammalla laitealustalla yhtaikaa ja ohjelmistoalusta monistaa itsensä näille kaikille jakaen tehtävänsä eri laitealustojen kesken. Näin kokonaisia laitealustoja voi olla poissa toiminnasta sen haittaa-

matta kuitenkaan itse ohjelmiston suorittamista. Ylläpidon kannalta on kuitenkin vaikeaa seurata, millä tietyllä laitealustalla tiettyä ohjelmistoa kulloinkin suoritetaan.

## 3.6 Huolto

Aktiivisessa käytössä olevan tietojärjestelmän laite- ja ohjelmistoalusta vaatii säännöllistä huoltoa, jotta käytettävyys- ja saatavuusodotukset voidaan pitää sopimusten edellyttämällä tasolla. Huollon ajankohdat tulevat eteen väistämättä, vaikka niitä ei ennalta määrättäisikään, mutta se heikentää järjestelmän imagoa asiakkaan näkökulmasta ja aiheuttaa sanktioita toimittajalle.

White kumppaneineen suosittelee pitämään kriittisiksi havaituista komponenteista varakappaleita ylläpidon saatavilla mahdollisen ongelman nopeaa korjaamista varten sekä ennaltaehkäisevänä työnä puhdistamaan ja tarvittaessa vaihtamaan järjestelmän kuluvia komponentteja, kuten tuulettimia, jotta toiminnallisien käyttökatkoksien määrää pystyttäisiin minimoimaan [22].

Suunnitelmallisella huoltostrategialla ohjelmistoalustan ohjelmistot voidaan asiakkaan antaman hyväksynnän mukaisesti huoltaa ja laitealustan kriittisiä komponentteja korvata tai uudistaa siten, että asiakkaan toimintaa häiritään mahdollisimman vähän. Näin saadaan minimoitua havaittujen ongelmatilanteiden kehittyminen ja koordinoidusti korjattua järjestelmässä havaittuja puutteita.

## 3.7 Päivitykset

Tietojärjestelmä kehittyy alati uusien innovaatioiden johdosta ja sen suorituskykyä, tietoturvaa ja luotettavuutta saadaan asteittain nostettua. Tällaisia muutoksia ei kuitenkaan automaattisesti oteta järjestelmissä käyttöön, vaan näiden vaikutusta järjestelmään kokonaisuutena punnitaan tarkasti ennen uudempaan kehitysversioon siirtymistä.

Uusien päivitysversioiden hankkiminen tietojärjestelmään tulee suorittaa harkitusti, sillä on punnittava uuden version tuomat hyödyt tämän muutoksen aiheuttamiin kustannuksiin verrattuna. Tietojärjestelmän muuttamisessa on myös muistettava, että tämä saattaa aiheuttaa muutospaineita myös ajettavissa ohjelmistoissa, jolloin muutoskustannukset versiosta toiseen nousevat moninkertaisiksi.

White kumppaneineen tähdentävät onnistuneen järjestelmän päivittämisen koostuvan seuraavista osavaiheista[22]:

- Yhtaikaisten muutoksien lukumäärä kannattaa minimoida
- Muutoksia kannattaa testata ensin pienessä osassa järjestelmää ennen laajamittaista muutostyötä
- Muutoksien tekeminen ja mahdollinen palautuminen aiempaan versioon kannattaa dokumentoida huolellisesti

- Järjestelmän toiminta on hyvä tarkistaa uuden järjestelmän käynnistyksen yhteydessä tai välittömästi sen jälkeen

Whiten ja kumppaneiden tutkimuksen tuloksena syntyneiden yleispätevien ohjeiden taustalla ovat olleet tarve vähentää käytettävän järjestelmän seisokkiaikaa, helpottaa ongelmalähteen löytämistä ja tehostaa ylläpidollista kommunikointia. Uusien ominaisuuksien lisäämisellä vähitellen kokonaisuuteen yksinkertaistaa mahdollisten komplikaatioiden havaitsemista ja ratkaisemisprosessia. Uudet ominaisuudet on hyvä testata pienemmässä mittakaavassa ja dokumentointi on hyvä pitää ajantasalla, jotta mahdollisten ongelmien ilmaantuessa palautuminen aiempaan toimineeseen konfiguraatioon ei muodostu ylitsepääsemättömän vaikeaksi ja järjestelmä kyetään saamaan toimintakuntoon mahdollisimman lyhyessä ajassa.

## 3.8 Tietoturva

Tietoturvan takaaminen on ylläpidon kannalta yhtä tärkeässä roolissa kuin koko tietojärjestelmän toimivuus yleensä. Tietojärjestelmän käytettävyyden arvo laskee, jos havaitaan tietojärjestelmän omaavan tietoturvallisuuteen liittyviä haavoittuvaisuuksia, jotka avaavat mahdollisuuden kolmannelle osapuolelle kerätä strategisesti merkittäviä tietoja joko tietojärjestelmästä itsestään tai siihen kohdistuvasta tiedonsiirrosta. Devanbun ja Stubblebinen mukaan tietoturvaa ei kuitenkaan kaikesta huolimatta oteta huomioon tarvittavissa määrin tietojärjestelmien ohjelmistoissa, vaan tietoturvaa lisätään jälkikäteen tehtyyn ohjelmistoon [4].

Tietoturvaa uhkaavia potentiaalisia riskejä on olemassa lukuisia, eikä niitä kaikkia vastaan pystytä ohjelmallisesti varautumaan. Tietoturvaratkaisuja tarvitaan etenkin tietojärjestelmän toimivuuden takaamiseksi, tietojen turvaamiseksi, tiedonsiirron varmistamiseksi ja käyttäjien tunnistamiseksi. Nykyisillä teknologioilla voidaan tiedon tallennusta turvata kryptaamalla tiedot tallennusmedialla, tiedonsiirtoa voidaan salata salausavaimin ja tietojen muokkaaminen ja käsittely voidaan rajata ainoastaan tiettyjen järjestelmän käyttäjien oikeudeksi, mutta etenkin inhimillisten tekijöiden poissulkeminen on käytännössä mahdotonta. Kuten Turban kumppaneineen toteaa, ihmisillä on tapana laiminlyödä turvallisuusmääräyksiä, koska kokevat ne epämiellyttäviksi [19].

Ohjelmiston tuottajan ja asiakkaan välinen ylläpitosopimus kattaa yleensä tietoturvaan liittyvät päivitykset takuuaikana. Ylläpitosopimus määrittää ohjelmistoalustan ylläpitoa pitämään tietoturva-asetukset ajan tasalla, jottei tietoturvamurtoa tapahtuisi ohjelmistoalustassa havaitun tietoturvariskin avulla.

Tietoturvaa kokonaisuutena suunniteltaessa on tärkeää selvittää haluttu turvallisuustaso asiakkaan kanssa, jotta saavutettu turvallisuus ja niistä aiheutuvat kustannukset ovat asiakkaan puolesta hyväksyttäviä. Turbanin ja kumppaneiden mukaisesti tärkeitä osa-alueita tietoturvan puolustusstra-

tegian määrittämisen kannalta ovat ennaltaehkäisy, tietoturvarikkomuksen havaitseminen, vahingon rajoittaminen, toiminnan palautuminen ja korjaaminen [19].

### 3.8.1 Ennaltaehkäisy

Ennaltaehkäisyllä tarkoitetaan sitä, että tietojärjestelmää toteutettaessa ja sitä ylläpidettäessä rakennetaan tiedonsiirto, tiedon tallennus ja käyttäjäautentikointi siten, että tiedon joutuminen kolmannelle osapuolelle on hankalaa, miltei mahdotonta. Ennaltaehkäiseminen on ratkaisevassa asemassa järjestelmän kokonaisturvaa tarkasteltaessa.

### 3.8.2 Havaitseminen

Havaitsemisella tarkoitetaan, että tietojärjestelmää suunniteltaessa ja sitä ylläpidettäessä on otettu huomioon kuinka voidaan havaita tilanteet, jolloin on tapahtunut mahdollinen tietomurto. Yläpidossa näin järjestelmän tilaa seuraamalla havaita väärinkäytökset ja ryhtyä tarvittaviin vastatoimenpiteisiin vahinkojen minimoimiseksi.

### 3.8.3 Rajoittaminen

Rajoittamisella tietoturvamielessä tarkoitetaan tappioiden rajoittamista tapauksessa, jossa järjestelmä on haavoittunut tietoturvamurron yhteydessä. Eli toisin sanoen rajoittamisessa pyritään järjestelmä saamaan takaisin käyttökuntoon mahdollisimman lyhyessä ajassa ongelman ilmentymisestä.

### 3.8.4 Palautuminen

Palautumisella tarkoitetaan niitä toimenpiteitä, joiden avulla järjestelmä palautetaan vikatilaa edeltäneeseen toimivaan tilaan mahdollisimman nopeasti. Tällaisessa tilanteessa mahdollisia toimintamalleja on joko korjata vanha järjestelmä kuntoon tai korvata järjestelmällä uudella. Järjestelmän kriittisyys toiminnan kannalta asettaa ehdot sille, kumpaan ratkaisuun päädytään.

### 3.8.5 Korjaaminen

Korjaamisella tarkoitetaan niitä toimenpiteitä, joiden avulla pyritään ehkäisemään havaitun vikatilanteen uusiutuminen tulevaisuudessa. Näin pyritään korjaamaan suunnittelussa tehtyjä virheolettamuksia ja parantamaan järjestelmän vakautta jatkossa.

## 3.9 Yhteenveto

Ehkäisevän ylläpidon tehtävänä on tukea tietojärjestelmän toimintaa siten, että ei-toivottu toiminnan estyminen saataisiin minimoitua niin käyttäjien opas-

tamisen kuin järjestelmän huoltamisen, päivittämisen ja valvonnan avustuksella.

Suoritettavalla ohjelmistolla on hyvin harvoin suoraa yhteyttä tiedonkäsittelylaitteistoon, joten sitä käytetään yleisesti ohjelmistoalustan tarjoamien rajapintojen avulla. Ylläpidon näkökulmasta tiedonkäsittelylaitteiston ylläpitäminen vaatii jatkuvaa seurantaa, sillä fyysiset ongelmat aiheuttavat väistämättä häiriöitä suoritettaville ohjelmille. Laite- ja ohjelmistoalustan hajauttamisella aikaansaadaan vakaampia järjestelmiä, mutta niiden ylläpitäminen muuttuu haasteellisemmaksi kokonaisuuden hajautuessa useammalla tasolla rinnakkaisiin itsenäisesti toimiviin komponentteihin.

# Luku 4

# Hajautetun ohjelmisto- ja laitealustan valvonta

Tämän luvun tarkoituksena on syventyä tietojärjestelmän ylläpidossa hajautetun ympäristön tuomiin haasteisiin ja tämän lopputyön varsinaiseen oman työn osuuteen eli automatisoituun hajautetun tietojärjestelmän valvontajärjestelmään ja tämän kehitykseen johtaneisiin taustatekijöihin.

Hajautetussa ohjelmisto- ja laitealustassa tuotettavan palvelun vakauden takaamiseksi järjestelmän tehtävien prosessointia jaetaan usean rinnakkaisen osakomponentin kesken, jolloin nämä pystyvät korvaamaan toisiensa toimintoja järjestelmän logiikan avulla. Yksittäisten komponenttien viottuminen ei vaikuta itse palvelun tuottamiseen, sillä palvelun tuottaminen voidaan ohjata järjestelmän toimiville komponenteille.

Tietojärjestelmän hajauttaminen suunnittelemalla tämän rakenne mahdollisimman modulaariseksi niin ohjelmistollisella- kuin laitetasollakin tekee siitä ylläpidollisesti normaalia tietojärjestelmää helpomman ylläpitää ja samalla sen laajentamismahdollisuudet ja päivitettävyys paranevat, toteaa Crichlow hajautettuja tietojärjestelmiä käsittelevässä kirjassaan [13]. Pienimuotoisia tietojärjestelmiä ei ole kannattavaa toteuttaa hajauttamalla sitä osakomponentteihin, sillä hajautetun järjestelmän toteuttaminen ja suunnitteleminen on tavallista järjestelmää monimutkaisempi operaatio. Suuremmissa ohjelmistoprojekteissa hajauttamisen tuomat hyödyt ajavat toteutusteknisten haasteiden ohitse, sillä järjestelmää pystytään hallinnoimaan pienemmissä osakokonaisuuksissa. Vikatilanteet voidaan yksilöidä tiettyyn järjestelmän osaan ja näinollen järjestelmän monitorointi yksinkertaistuu ja ongelmatilanteiden selvittämistyö voidaan kohdistaa tarkemmin juuri kyseiseen ongelma-alueeseen. Tietojärjestelmän hallinnoinnissa on kuitenkin ymmärrettävä hajautetun tietojärjestelmän merkitys kokonaisuuden toimivuuteen, sillä eri osakokonaisuuksien vaikutus toisiinsa on pystyttävä luotettavasti todentamaan ja näitä pitää pystyä myös kontrolloimaan.

Hajautuksella suurikin tietojärjestelmä pystytään jakamaan pienempiin osakokonaisuuksiin eli komponentteihin ja niitä voidaan kehittää ja huoltaa

toisistaan riippumattomasti edellyttäen, että suunnittelussa eri komponenttien rajapinnat on määritelty kattavasti.

## 4.1 Taustaa

Hajautettu tietojärjestelmä koostuu useammasta komponentista, joiden ei välttämättä tarvitse olla keskenään samankaltaisia. Nämä komponentit sijaitsevat eri ajoympäristöissä ja ne kommunikoivat keskenään yhteisen kommunikointikanavan, kuten verkon, välityksellä. Yleensä hajautetussa tietojärjestelmässä tieto on jaettu koko järjestelmän kesken, mutta toiminta on jaettu erillisille itsenäisille yksiköille, kuten kuva 3.2 havainnollistaa.

### 4.1.1 Hajautusteknologiat

Hajautettuja järjestelmiä tukevia teknologioita on muutamia ja CORBA hajautusstandardin mukaisia ohjelmointikieliriippumattomia toteutuksia on miltei jokaiselle käyttöjärjestelmälle, toteaa Lahtela [10]. Java-ohjelmointikielellä toteutettuja hajautusmekanismeja voidaan toteuttaa Javan RMI-rajapinnan avulla, EJB-teknologialla tai jo aiemmin mainittua CORBA hajautusstandardia hyödyntäen. Myös Microsoftilla on omat teknologiansa hajautuksen toteuttamiseen, mutta näitä ei käsitellä tämän lopputyön puitteissa.

### 4.1.2 Toteutusmallit

Yleisesti hajautetun sovelluksen toteuttamisessa hyödynnetään arkkitehtuurisuunnittelussa niin sanottuja toteutusmalleja, joiden avulla toteutettava järjestelmä voidaan jakaa useampaan tasoon ja niiden suunnittelua voidaan jatkaa pienemmissä osissa. Toteutusmalleiksi käsitetään yksi-, kaksi-, kolme- ja monitasomallit. Yksitasoisella toteutusmallilla tarkoitetaan keskitettyä yritysjärjestelmää, jossa kaikki toiminnot on toteutettu samalla palvelimella ja sitä käytetään yksinkertaisen päätteen avulla, joskaan yksitasomallia ei voida hyödyntää hajautettua järjestelmää kehitettäessä. Kaksitasomalli tunnetaan myös asiakas/palvelin-mallina, jolloin järjestelmää ajetaan myös asiakaskoneella yksitasoiseen toteutukseen verrattuna. Kolmitasomallissa ideana on jakaa koko järjestelmä kolmeen osaan: käyttöliittymään, liiketoimintalogiikkaan ja tiedonhallintaan. Monitasomallissa järjestelmä jaetaan kolmea useampaan loogiseen osioon, jotka kaikki toimivat itsenäisesti ja kommunikoivat toisien osioiden, komponenttien, kanssa määriteltyjen rajapintojen välityksellä.

### 4.1.3 Väliohjelmistot

Väliohjelmistot näyttelevät tärkeää osaa hajautettujen järjestelmien toteutuksessa, sillä ne toteuttavat suurimman osan monitasoisten hajautettujen järjestelmien keskeisistä palveluista. Yleisesti muun muassa käyttäjähallintaan,

tietokannanhallintaan, klusterointiin, monitorointiin ja tietoturvaan palveluita tarjoavat sovelluspalvelimet ja komponenttialustat. Hajautettua ohjelmistoa toteutettaessa otetaan yleensä tällainen väliohjelmisto yksinkertaistamaan toteutusta ja selkeyttämään toteutettavan ohjelmiston ylläpitoa. Väliohjelmisto toimii ohjelmistolle rajapintana laitealustaan, joka voidaan väliohjelmistoa hyödyntäen hajauttaa useampiin eri osakokonaisuuksiin ja toteutettu ohjelmisto käyttää tätä kuin normaalia laitealustaa. Tällaisella modulaarisella ohjelmointikäytännöllä yksinkertaistetaan toteutettavan ohjelmiston rakennetta ja voidaan nopeuttaa ohjelmistoprojektin toteutusaikataulua valmisohjelmia hyödyntämällä.

### 4.1.4 Valvonta

Hajautetun järjestelmä on valvonnan kannalta kaksijakoinen, sillä toisaalta järjestelmän jakaminen komponentteihin avaa ylläpidolle mahdollisuuden nähdä kunkin erillisen komponentin kulloisenkin tilan, mutta toisaalta seurattavien komponenttien määrä kasvaa pikaisesti järjestelmän koon kasvaessa sellaisiin mittoihin, että kokonaisvaltaisen kuvan saamista järjestelmän toiminnasta on hankalaa saada aikaan.

Siinä missä yksi-, kaksi- tai kolmitasoinen toteutusmallin valvonnassa mahdollinen vikatilanne voidaan havaita monitasoisen toteutusmallin vastaavaa tilannetta nopeammassa ajassa, on monitasoisen toteutusmallin etuna, että vika voidaan paikantaa tarkasti tiettyyn järjestelmän osaan ja koska monitasoisessa toteutusmallissa komponentit toimivat toistensa kanssa rajapintojen välityksellä, voidaan tämä viallinen komponentti korvata nopealla aikataululla toimivaan versioon ja kuitenkin tänä aikana järjestelmän muut osat toimivat häiriöttä.

## 4.2 Tavoitteet

Tämän lopputyön tavoitteena on kehittää hajautetun ohjelmisto- ja tietojärjestelmän valvontaa automatisoiden tiedon keräystä ja jäsentää eri komponenteilta saatavaa valvontatietoa kokonaisuudeksi siten, että järjestelmän kriittiset toiminnat pystyttäisiin valvomaan yhtä hallintaliittymää apuna käyttäen. Valvonnan automatisoinnin haasteena on eriyttää valvontaelin riittävän erilliseksi itse valvottavasta järjestelmästä, jotta minkään yksittäisen komponentin vioittuminen ei häiritse valvontaa, mutta kuitenkin jokaisesta toimivasta komponentista saadaan kerättyä valvontatietoa ylläpidolle ja myös viittaukset mahdollisesti toimimattomiin komponentteihin.

Hajautetun valvontatiedon yhdistämisellä ja esikäsittelyllä voidaan minimoida ylläpitoon kuluvaa aikaa ja ylläpidollisiin toimiin pystytään ryhtymään lyhyemmässä ajassa verrattaessa tilanteeseen, jossa jokaista komponenttia valvottaisiin erillisesti. Mahdollisen ongelmalähteen paikantaminen yksinkertaistuu valvontatietojen esikäsittelyllä lyhentäen palvelulle aiheutuvia häiriöitä. Valvontaohjelman konfiguroinnissa on eri komponenttien hälytystaso-

jen määrittäminen tärkeässä asemassa, sillä tämän jälkeen valvontaohjelmiston vastuulle voidaan jättää ongelmatilanteista hälyttäminen ylläpidolle, joskin täysin automatisoitua hälytysjärjestelmästä ei voida tehdä, sillä myös valvontajärjestelmässä voi esiintyä ongelmatilanteita. Valvontatiedon keräämisellä voidaan pidemmän aikajakson kuluessa aikaansaada ennustettavuutta järjestelmän käyttäytymiseen.

Kontrolloidun valvonnan ja aktiivisen seurannan avulla pyritään parantamaan asiakkaiden kokemaa kokonaispalvelua ja kohentamaan tätä kautta asiakastyytyväisyyttä. Valvontatiedon tulkitsemisella voidaan myös keskittää kehitystyötä järjestelmän kriittisiin osa-alueisiin.

## 4.3 Kaupallinen motivaatio

Hajautetun ohjelmiston valvonnan automatisoimisesta on hyötyä niin asiakkaan kuin toimittajankin näkökulmasta katsottaessa. Tarjottavana oleva ohjelmisto saadaan aktiivisen valvonnan avulla reagoimaan vikatilanteisiin nopeammin kuin mihin päästään normaalin valvonnan avulla ja kuormituksen hienosäätäminen eri osakokonaisuuksien kesken pystytään suorittamaan automaattisesti kerätyn tiedon perusteella maksimaalisen suorituskyvyn aikaansaamiseksi.

Asiakkaan kannalta jo ohjelmiston hajauttaminen aikaansaa siitä vakaamman, luotettavamman ja sen saatavuus paranee verrattaessa normaaliin yhdellä laite- ja ohjelmistoalustalla toimivaan ohjelmistoon. Hajautetun järjestelmän automaattisella valvonnalla pystytään reagoimaan nopeammassa aikataulussa järjestelmässä havaittuihin vikatilanteisiin ja mahdolliset käyttökatkokset voidaan minimoida tai poistaa miltei kokonaan ohjaamalla suoritusta toimivalle osalle järjestelmää samalla kun viallista osaa järjestelmästä korjataan. Valvonnan avulla voidaan myös jakaa palvelun aikaansaamaa kuormitusta siten, että käytössä olevista resursseista saadaan maksimaalinen tuotto aikaan ja tällä tavalla parannetaan palvelun tuottavuutta. Näillä osa-alueilla valvonnan automatisoinnilla saavutetulla kehityksellä parannetaan palvelua käyttävien asiakkaiden asiakastyytyväisyyttä, jolloin ohjelmiston julkinen kuva saa positiivista nostetta ja uusien asiakassuhteiden solmiminen helpottuu positiivisen maineen myötä.

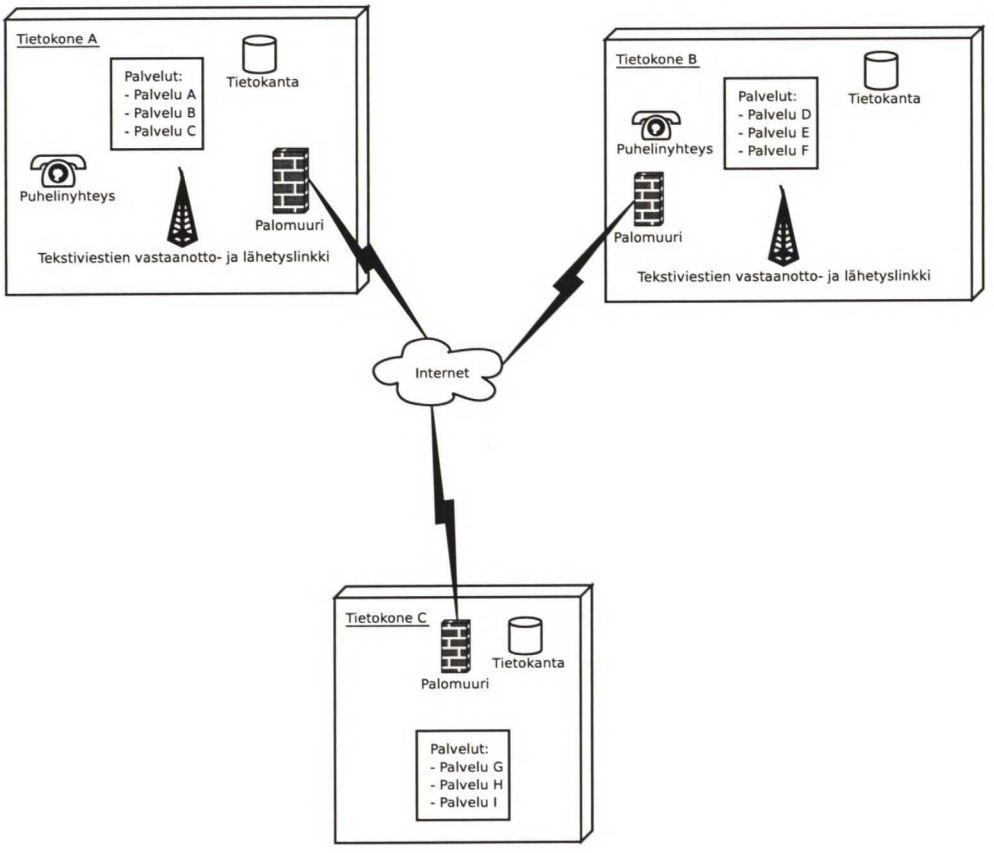
Ohjelmiston valvonnan automatisoiminen vähentää tarvittavia ylläpidollisia resursseja ja täten aikaansaa ylläpidosta vastaavalle taholle taloudellisia säästöjä erityisesti henkilöstökuluissa, sillä lähes kaiken henkilöstöresurssin voi koordinoida itse ongelmien korjaamiseen ja normaalissa tilanteessa ylläpidolliset vaateet jäävät vain ylläpito-ohjelmiston toimivuuden tarkistamiseen. Näiden resurssisäästöjen avulla voidaan joko kehittää ohjelmistoa entisestään, panostaa laitteistohierarkiaan aikaisempaa enemmän tai vaihtoehtoisesti laskea hajautetun ohjelmiston hankinta- ja ylläpitokustannuksia saaden uusia asiakkaita kiinnostumaan ohjelmistosta ja aikaisemmat asiakkaat pitäytymään ohjelmiston käyttäjinä.

Automaattisen valvonnan myötä myös ohjelmiston skaalautuvuus suuremmalle käyttäjämäärälle yksinkertaistuu, sillä kokonaisuuden kasvaessa ylläpidollisen valvontakuormituksen kasvu voidaan pitkälti hoitaa ohjelmallisesti ylläpidosta vastaavan tahon selvitessä tehtävistään aiemmalla henkilöstömäärällä.

## 4.4 Edeltävä tilanne

Järjestelmä kokonaisuudessaan on ollut ennen tämän tutkimuksen alkua riittävä pienelle asiakaskunnalle ja vähäiselle kuormitukselle. Kasvavan asiakasmäärän myötä on kuitenkin nykyinen toteutusmalli havaittu heikosti skaalautuvaksi ja ongelmatilanteiden havainnointi on hankalaa valvontatiedon ollessa hajallaan eri palvelinkonfiguraatioissa ja palveluissa. Ohjelmistossa tai laitteistossa tapahtuva häiriö johtaa vähintään yhden ohjelmiston toimimattomuuteen ja useissa tapauksissa häiriö vaikuttaa ohjelmistoalustan kautta useisiin ohjelmistoihin ja näin palvelimen ohjelmistot voivat lyhyessäkin ajassa lopettaa niille tarkoitetun prosessoinnin. Yksittäisen ohjelmiston häiriötilanteesta lähetetään järjestelmän ylläpidolle informaatiota tekstiviestein tai sähköpostin avulla, mutta kuormituksesta johtuvia palvelun hidastumisia tai fyysisten resurssien loppumisesta on ylläpidon vaikeaa saada reaaliaikaista tietoa. Näinollen onkin mahdollista, että palvelua käyttävä asiakas on ensimmäinen henkilö, joka havaitsee ongelmatilanteen käyttämässään palvelussa ja tämä tapa ei ole suotavaa kestävien asiakassuhteiden eikä hyvän yrityskuvan luonnissa.

Tämän hetkinen järjestelmän arkkitehtuuri on toteutettu siten, että jokaisella laitealustalla on oma yhteys Internetiin, oma erillinen tietokantansa, omat erilliset palvelunsa sekä muutamalla palvelimella on myös oma puhelinliittymä tiedonvälitykseen. Tätä konfiguraatiota selventää kuva 4.1.




Kuva 4.1: Aikaisempi laitealustan toteutus

Yksittäinen laitealusta suorittaa sille asennettuja ohjelmistoja yksin käyttäen omaa tietokantaansa ja omia tietoliikenneyhteyksiään. Häiriötilanteesta ajettavassa ohjelmistossa aiheutuu väistämättä ainakin tämän kyseisen ohjelmiston tarjoaman palvelun keskeytyminen ja pahimmassa tapauksessa kaikkien tarjottavien palveluiden estymisen. Vastaavan ongelman aiheuttaa myös laitealustassa ilmenevä vika. Ongelman havaitsemisen ja korjaamisen ajan häiriöstä kärsivät palvelut ovat estyneitä toimimaan, sillä ne ovat konfiguroitu toimimaan juuri tällä laite- ja ohjelmistoalustalla, eikä niiden siirtäminen tilapäisesti toiselle palvelimelle ole yksinkertaisesti toteutettavissa.

Ohjelmistot on toteutettu pääosin hyödyntämällä sovelluspalvelinteknologiaa, joten toteutusmalliltaan ohjelmistot ovat monitasoisia ja omaavat näinollen mahdollisuuden hajautusteknologian käyttämiseen. Hajautusta ohjelmistoeikä laitealustatasolla ei olla kuitenkaan tässä vaiheessa toteutettu kustannuskysymysten vuoksi.

Tietoliikenneyhteyksissä ja sähkönsaatavuudessa ilmenevät ongelmat aiheuttavat käytännössä koko palveluntarjonnan lamautumisen, sillä vaikka itse palvelinlaitteet ovatkin suojattuja sähkökatkosten varalta, katkeaa tietoliikenneyhteydet ja tämän myötä palveluiden käyttö estyy täysin. Palveluiden saatavuuden takaamiseksi olisikin syytä pohtia mahdollisuuksia Internetyhteyksien kahdentamiseen.

Järjestelmän keskeisiä puutteita näinollen ovat laitealustan ja internetyhteyksien kahdentamattomuus, kuormituksen jakamisen uupuminen laitealustojen kesken ja valvontatiedon hajanaisuus ja sen mukana tuoma heikko toiminnan yleiskuvan hahmottaminen.

Huomioitavaa tässä vaiheessa on, että tässä lopputyössä ei pyritä kuitenkaan parantamaan nykyisen järjestelmän toimivuutta eikä toimintavarmuutta, vaan pyritään kehittämään automaattinen valvontajärjestelmä, joka pystyy valvomaan tietojärjestelmää sen fyysisestä rakenteesta riippumatta. Nykyinen tietojärjestelmä vaatii osakseen kehitystyötä, mutta tämä jääköön tulevaisuuden kehityskohteeksi.

## 4.5 Yhteenveto

Automaattisen valvontajärjestelmän toteuttamisella kykenee keräämään hyvin erityyppisten ohjelmisto- ja laitteistoratkaisujen tilatietoja ja koostamaan näistä ylläpidollisesti merkittävää kokonaiskuvaa yleisestä toimivuudesta. Valvonnan yhtenäistämisellä ja automatisoinnilla voidaan tehostaa hajautuksesta johtuvan kasvaneen tietomäärän hallintaa.

Lopputyön omantyön osuutena toteutettavan valvontasovelluksen tarkoituksena on kerätä informaatiota nykyisen laite- ja ohjelmistoalustan tilasta ja yhdistää niitä siten, että ongelmatilanteet voidaan havaita tehokkaammin ja niitä voidaan mahdollisesti jopa ennustaa ennakkoon.

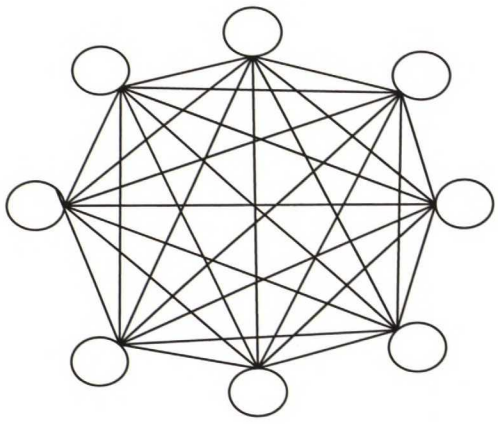
# Luku 5

# Automaattisen valvontasovelluksen määrittely

Valvontamekanismin suunnittelu hajautetulle tietojärjestelmälle siten, että valvontatieto on kaiken aikaa ylläpitäjän saatavilla ja ongelmakohdat löytyvät selkeästi hallintaliittymän avulla, on toteutusteknisesti haastava prosessi. Pohdinnan arvoisia kohteita ovat tiedonsiirron toteuttaminen, tietojen reaaliaikaisuus, valvontajärjestelmän toimivuuden takaaminen tietojärjestelmän vioista huolimatta ja mahdollisien ongelmatilanteiden ennustaminen edeltäkäsin. Tämä kappale tulee keskittymään sovelluksen tiedonsiirtomekanismiin, rakenteeseen, tietoturvaan ja tietojen esitystavan määrittelyyn.

## 5.1 Tiedonsiirto ja valvontaverkoston rakenne

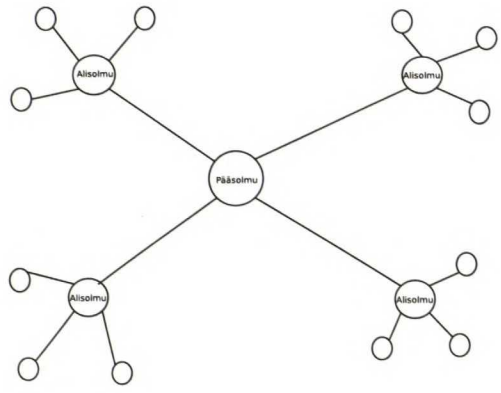
Jaetun tiedon maksimoimiseksi jokaisen hajautetun tietojärjestelmän valvonnan osakomponentin kesken jokaisen valvonnan osakomponentin olisi välitettävä omat tietonsa jokaiselle muulle valvonnan osakomponentille. N komponenttia sisältävän verkon jokaisella komponentilla olisi näinollen N-1 muuta komponenttia, jolle tämän tulisi tietonsa välittää. Vaikka yksittäisen komponentin välittämät tietomäärät jäävät vähäisiksi komponentin tilasta muille raportoitaessa, muodostuu ongelmaksi tiedonsiirron ruuhkautuminen lukuisien eri komponenttien vaihtaessa tietoja keskenään. Tällaista verkkotopologiaa kutsutaan Mesh-verkoksi eli jokainen verkon komponentti on suorassa yhteydessä jokaisen muun komponentin kanssa. Tästä havainnoillistavana kuvana 5.1.



Kuva 5.1: Mesh-verkko

Tiedonsiirron hallinnan parantamiseksi on kannattavaa ryhmittää tiedonvälitystä siten, että tietojärjestelmän valvonta jaetaan loogisiin osakokonaisuuksiin, joissa jokaisessa on yksi pääkomponentti osakokonaisuuden valvontatietoa keräämässä ja tämä välittää valvontatiedot hierarkiassa seuraavaksi ylemmälle tasolle ja näin tieto kulkeutuu viimein koko hajautetun tietojärjestelmän valvontaa hoitavalle pääkomponentille. Tällaisesta toteutustavasta käytetään nimitystä hierarkinen tähti, josta selventävänä kuvana 5.2. Kuvan mukaisesti pääsolmuun liittyy N kappaletta alisolmuja ja kukin alisolmu voi toimia pääsolmuna siihen liittyville M kappaleelle alisolmuja ja niin edelleen.

Valvontajärjestelmä, jonka kaikki osakomponentit käsittelevät samaa tietoa, on läheisesti synkronisoitu, mutta jos ainoastaan yksi ryhmän jäsen käsittelee saatavissa olevaa tietoa ja välittää tuottamansa tulokset muille järjestelmän jäsenille, puhutaan väljästi synkronoidusta järjestelmästä. Cristian kumppaneineen huomauttaa, että jo suunnitteluvaiheessa on tärkeää päättää synkronisoinnin taso järjestelmälle asetettujen vaatimusten mukaan [3]. Näin ollen tämän valvontajärjestelmän osalta tiedonsiirron kontrolloimiseksi ja verkkoliikenteen minimoimiseksi päädyttiin väljään synkronointimalliin, jotta valvontaan kulutettu tiedonsiirtokapasiteetti ei häiritsisi tarpeettomasti muita tuotantosovelluksia.

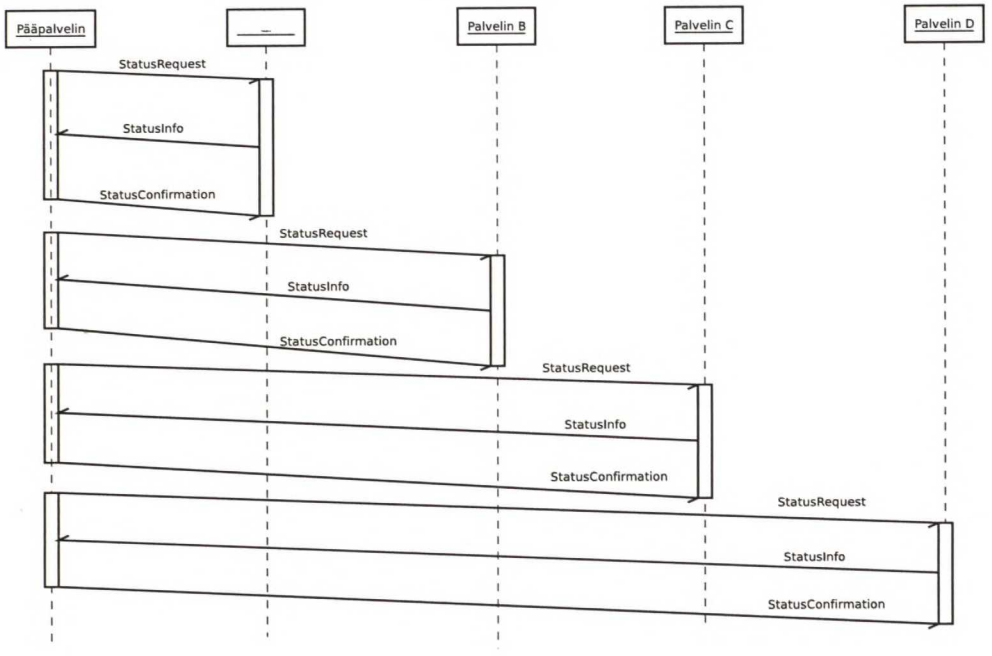



Kuva 5.2: Hierarkinen tähti

Myös tähtihierarkiaa verkon rakenteena hyödynnettäessä on pohdittava kommunikointiproseduurin logiikkaa eri komponenttien kesken, sillä vaikka on sovittuna, että ainoastaan verkon päätason komponentti prosessoi verkon tilatietoa, saattavat verkon alemman tason komponentit ruuhkauttaa komponentin prosessointia syöttämällä tälle kapasiteettin nähden tarpeettoman paljon verkon tilatietoja. Tällaisen ongelmatilanteen voi ratkaista määrittämällä, että tiedot välitetään alemman tason komponentilta ylemmälle tasolle vasta ylemmän tason pyytäessä komponentilta tämän tietoja.

### 5.1.1 Tilatiedon hankkiminen alikomponentilta

Tiedonsiirron minimoimiseksi osakomponenttien kesken on myöskin perusteltua suorittaa tiedonsiirtoa ainoastaan pääsolmun ja alisolmun kesken. Samalla tasolla olevien komponenttien välinen viestiliikenne on käytännössä tarpeetonta tietojen toistoa, jonka vuoksi sitä kannattaa välttää mahdollisuuksien mukaan. Valvontaverkossa tapahtuvan häiriötilanteen korjaamiseksi voi kuitenkin olla perusteltua vaihtaa tietoja myös saman tason komponenttien kesken, joten erityistapauksissa tämäkin tulee sallia. Tiedonsiirtotapahtuman tulisi tapahtua pääsolmun herätteestä, jolloin tiedon kerääminen alisolmuilta on kontrolloitua ja tarpeeton viestinvälitys saadaan minimoitua. Tiedonsiirtotapahtuma pääsolmun näkökulmasta on esitetty asiankäsittelydiagrammissa 5.3.




Kuva 5.3: Tiedonsiirtotapahtuma pääsolmun näkökulmasta

Tiedonsiirtotapahtuma pääsolmun ja sen alisolmun kesken jaetaan kolmeen osaan: tietojen pyytäminen alisolmulta, tietojen vastaanottaminen alisolmulta ja tietojen vastaanottamisen kuittaus alisolmulle, joka pitää sisäl-

lään myös päivitettyä tietoa tietojärjestelmän kokonaistilasta.

Tietojen pyytäminen alisolmulta indikoi alisolmulle, että on tämän vuoro välittää tietonsa eteenpäin. Näin vältytään turhilta tietojen lähetysyrityksiltä silloin, kun niiden vastaanottamisen käsittelyyn ei olla varauduttu. Tietojen vastaanottamisessa pääsolmu päivittää kyseessäolevan alisolmun tiedot tietokantaan ja päivittämisen jälkeen kuittaa operaation päättyneeksi alisolmulle. Alisolmulle lähetettävässä kuittaussanomassa voi tarpeen mukaan lähettää myös muualta kerättyä valvontatietoa alisolmun käyttöön.

Menettelyllä, jossa yläkomponentti pyytää tietoja erikseen jokaiselta alikomponentilta ei tiedonvälitystä saada suoritettua minimaalisella sanomamäärällä verkkoliikennettä kokonaisuutena tarkasteltaessa, sillä alikomponenttien lähettäessä tietojaan säännöllisesti muuttuneen tilatiedon jäljiltä sanomaliikenteestä jäisi pois tietojen pyyntisanoma. Tällaisessa muutoksen myötä tapahtuvassa yläkomponentin informoinnissa rasitetaan kuitenkin jo sangen pienillä valvontaverkoilla tilatietoja käsittelevää verkon komponenttia suhteessa huomattavasti enemmän kuin pyyntipohjaisessa verkon valvonnassa.

## 5.2 Tietojen reaaliaikaisuus

Valvontatiedon välittyminen tarkkailtavasta kohteesta valvontaan olisi suotavaa olla reaaliaikaista, mutta reaaliaikainen valvontatiedon ylläpitäminen hajautetussa tietojärjestelmässä aiheuttaa tiedonsiirron ylikuormitusta eri osakomponenttien välittäessä tietoja häiriten mahdollisesti joidenkin tietojärjestelmän ohjelmistojen toimintaa, mikä ei ole suotavaa. Tämän johdosta jokainen valvonnan osakomponentti kerää oman osa-alueen valvontatietoa jokaiselta alikomponentiltaan vuorotellen, mutta välittää nämä ylemmälle taholle vasta niitä pyydettäessä. Valvontatietojen tiedusteluväli alikomponenteilta on hyvä asettaa suhteellisen pitkäksi, jotta tiedot ehditään keräämään ja käsittelemään verkon jokaisella tasolla. Noin minuutin välein suoritettava tilan tarkistus, jotta ilmenneeseen ongelmatilanteeseen ehditään reagoimaan palvelusopimuksen edellyttämällä tehokkuudella, on riittävä.

Valvontatiedon käsittelyn yhteydessä päätetään hälytyksen lähettämisestä ylläpidolle, joten valvontaverkossa havaituista ongelmatilanteista ylläpito saa tiedon noin minuutin viiveellä todellisesta tapahtumahetkestä. Jos valvontaverkko ei ole täysin vioittunut, pyrkii tämä korvaamaan vioittuneita komponentteja edellisessä kappaleessa esitettyjen operaatioiden avulla, jotta ylläpito saisi valvontaverkosta riittävästi informaatiota myös ongelmatilanteen selvityksen aikana.

## 5.3 Tietokannan toteutus

Valvontatietojen kerääminen vaatii tietokannalta ristiriitaisia ominaisuuksia, sillä tietokannan tulisi omata valvontatietoa pitkältä aikaväliltä, mutta silti tietojen hakemisen tulisi olla nopea operaatio. Valvontatietojen keräämi-

nen säännöllisesti lyhyin näytteenottovälein kerryttää tietokannan kokoa ja oleellisten tietojen hakeminen muuttuu entistä raskaammaksi. Silti myöskään vanhoja tietoja ei voi täysin poistaa tietyn aikarajan jälkeen, sillä tämä estää mahdollisuuden tulkita järjestelmän käyttäytymismalleja pitkällä aikajänteellä.

Tätä samaa ongelmaa on pohtinut myös Ahola diplomityössään [2], jossa hän on päätynyt harventamaan kerätystä informaatiosta mittaushetkestä kuluneen ajan perusteella osatuloksia pois. Näin hän on kyennyt säilyttämään niin pidempiaikaista tapahtumahistoriaa kuin antamaan tarkkaa tilastotietoa lähimenneisyydestä pitäen kuitenkin mittaustietokannan koon hallittavissa puitteissa. Ahola on päätynyt tietokannan mittaustietojen esiintymistiheyden säännöstelyssä viiteen tasoon, jossa ensimmäinen taso omaa viikon vanhan mittaustiedon minuutin mittausvälillä, toinen taso kuukauden päähän ulottuvan ajanjakson mittaustiedot kolmen minuutin mittausvälillä, kolmas taso kolmen kuukauden päähän ulottuvan ajanjakson mittaustiedot kuuden minuutin mittausvälillä, neljäs taso vuoden päähän ulottuvan ajanjakson mittaustiedot puolen tunnin mittaus välillä ja viides taso viiden vuoden päähän ulottuvan ajanjakson mittaustiedot tunnin mittausvälillä. Viittä vuotta vanhemmat tiedot Ahola poistaa tietokannasta kokonaan. Tällainen malli soveltuu myös toteutettavana olevan valvontaverkon mittaustietojen varastointiin. Aholan esittämää mittaustietojen harventamistapaa ja näin syntyvien mittaustietojen määrää selventää taulukko 5.1.

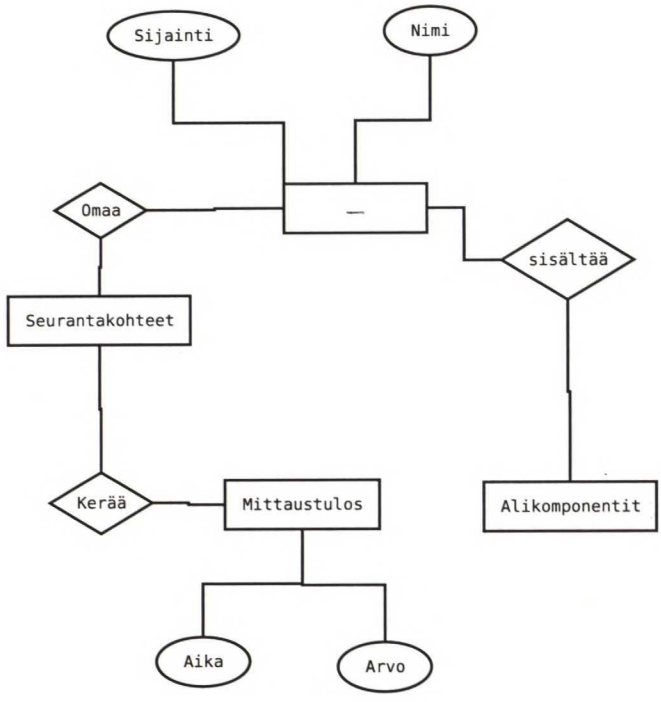
Taulukko 5.1: Mittaustietojen harventamisesimerkki [2]

| Taso | Aika | Mittausten väli | Rivien lukumäärä (noin) |
|---|---|---|---|
| Taso 1 | 0-1 vko | 1 min | 10000 |
| Taso 2 | 1 vko - 1 kk | 3 min | 10000 |
| Taso 3 | 1 kk - 3 kk | 6 min | 14000 |
| Taso 4 | 3 kk - 1 a | 30 min | 13000 |
| Taso 5 | 1 a - 5 a | 1 h | 35000 |
| Yhteensä | | | 83000 |

Tämä edellä mainittu Aholan ideoima malli tuo mittaustuloksiin tilastollista laajuutta, mutta tietoja poistettaessa ongelmana on näytteenottotavan merkitys tietokannassa säilyviin mittauksiin. Tietokantaan tallennetaan hetkellistä informaatiota järjestelmästä, jossa voi lyhyen ajan sisällä esiintyä suuriakin vaihteluja. Jos näitä mittaustuloksia poistetaan ainoastaan aikaleiman perusteella, saattaa järjestelmän aiemmasta käyttäytymisestä tietokannan perusteella muodostua hyvin erilainen kuva todellisuudessa tapahtuneeseen verrattuna. Esimerkiksi tästä ongelmasta voidaan ottaa muistinkäyttö valvontaverkon osakomponentissa: Komponentin muistikapasiteetti on ollut kokonaan käytettynä vajaan minuutin ajanjaksolla ja tästä on kirjautunut tieto tietokantaan. Kun tulee ajankohtaiseksi harventaa tietokannan tietosisältöä, jätetään mittauksista joka kolmas arvo jäljelle, jolloin voi tapahtua joko niin, että tulkitaan muistin olleen käytettynä koko ajanjakson ajan tai että muistia

ei olla juurikaan kuormitettu tuona ajanjaksona. Molemmat tulkinnat vääristävät todellista tilannetta huomattavasti ja tätä on syytä pyrkiä välttämään. Eräs toteutusvaihtoehto, jolla tätä ongelmaa pyritään kompensoimaan, on jättää tietokantaan mittaustuloksia harvennettaessa tämän harvennettavan mittaustulosjoukon keskiarvo. Näin tilastollisesti merkittävät vaihtelut näkyvät heilahteluina myös harvennetuissa mittauksissa.

Jokaisella valvontaverkossa toimivalla keskitason tai ylemmän tason komponentilla tulee olla luotuna tietokanta alikomponenttien tilanseurantaan, jolloin ylimmän toiminnassa olevan komponentin tiedoin pystytään esittämään tämän kattaman verkon osan aktiivista tilannetta muista valvontaverkon komponenteista riippumatta. Tietokannan tulee sisältää tiedot yksittäisen valvontaverkon komponenteista ja näiden kiinteistä ominaisuuksista, tiedot mitattavista suureista komponenteittain sekä ajan, komponenttitunnisteen ja tyypin mukaan indeksoidut valvontatiedot. Lisäksi valvontaverkoston hierarkiaa selventämään on hyvä koota tiedot komponenttien alikomponenteista. Tällainen tietokanta voidaan kuvata ER-kaaviona kuvan 5.4 mukaisesti.




Kuva 5.4: ER-kaavio tarvittavasta tietokannasta

## 5.4 Toimivuuden takaaminen

Valvontaohjelmiston haasteena on pystyä toimimaan ongelmitta hajautetun tietojärjestelmän ongelmatilanteista huolimatta. Tästä johtuen valvontajärjestelmän on varauduttava useampiin erilaisiin häiriötilanteisiin ja mukauduttava niihin valvonnan jatkuvuuden mahdollistamiseksi. Valvontaverkoston rakenteesta, joka määriteltiin kappaleessa 5.1, voidaan johtaa valvontaverkos-

ton koostuvan hierarkisesti kolmen tason osakomponenteista: Alin valvontaverkon taso, valvontaverkon keskitaso ja valvontaverkon päätaso.

Alimmalla valvontaverkon tasolla ei ole valvottavia alisolmuja, vaan se kerää tietoa vain omasta toiminnastaan ja välittää tiedot tilatietoja tiedustelevalle yläsolmulle. Alimman tason komponentin tulee varautua tilanteeseen, jossa komponentin välitön yläsolmu lopettaa häiriötilanteesta johtuen toimintansa. Tällaisessa tilanteessa tämän yläsolmun alaisten solmujen on valittava keskuudestaan yksi solmu korvaamaan viallinen yläsolmu, jotta hälytys- ja valvontatietojen kulku valvontaverkon päätasolle voisi jatkua myös ongelman havaitsemisen jälkeenkin.

Keskitason komponentin ongelmatilanne havaitaan ylemmällä verkon tasolla samoin kuin alimmankin tason komponentin ongelmatilanne, mutta tällaisessa tapauksessa tämän keskitason komponentin alaiset osakomponentit eivät kykene antamaan tilatietojaan ylemmälle tasolle. Ylläpidollisesti ajateltuna oleellista on toki saada tieto tästä ongelmallisesta komponentista, mutta toisaalta olisi informatiivista saada tietoa tämän komponentin alaisilta osakomponenteilta, sillä tämän avulla voidaan ongelman aiheuttaja rajata pienemmälle alueelle. Tällaisesta tilanteesta ongelmoivan komponentin välittömien alikomponenttien keskuudesta valitaan prioriteetiltaan korkein komponentti korvaamaan viallista keskitason komponenttia, joka informoi ylemmän tason komponenttia tapahtuneesta komponenttihierarkian muutoksesta ja tarvittaessa myös alempaa verkkohierarkiaa, jotta tapahtuneen muutoksen johdosta verkko vakiintuu toimivaan muotoonsa.

Ylimmän tason komponenttiin vaikuttavan ongelmatilanteen ilmetessä tämän välittömien alikomponenttien keskuudesta valitaan prioriteetiltaan suurin komponentti korvaamaan viallista ylimmän tason komponenttia ja uudeksi ylimmän tason komponentiksi valittu komponentti informoi aiempia alikomponenttejaan tapahtuneesta muutoksesta ja ryhtyy käsittelemään verkon valvontatietoja ja välittää ongelmatilanteista viestiä ylläpidolle.

Valvontaverkoston ylimmällä tasolla oleva komponentti ylläpitää tietoa valvottavan hajautetun järjestelmän tilasta ja prosessoi alisolmuilta saamaansa tilatietoa mahdollisien ongelmatilanteiden havaitsemiseksi. Ylimmän tason komponentti antaa ylläpidolle tiedot mahdollisista ongelmatilanteista. Ylläpidon informoiminen hälytyskanavaa pitkin voi myös vikaantua ja tällaisissa tapauksissa tulee valvontaverkostolla olla vaihtoehtoisia menetelmiä ylläpidon ajantasalle saattamiseksi. Ongelmaa ei pystytä välttämättä kiertämään, josta seuraa se, että ylläpidon on seurattava aktiivisesti hälytyskanavien toimintaa.

Ongelmatilanne voi näinollen kohdistua hajautetun tietojärjestelmän valvontaverkostossa taulukon 5.2 kuvaamiin osakomponentteihin.

Taulukko 5.2: Häiriötilanteen esiintymiskohteet ja mahdolliset ongelmat

| Ongelman esiintymiskohde | Esiintyvä ongelma |
|---|---|
| alin valvontaverkon osakomponentti | Tilatietoja ei olla kysytty tietyn aikajanan sisällä. |
| valvontaverkon keskitason osakomponentti | Joku valvottavista kohteista ei vastaa tilatiedusteluun tai tilatietoja ei olla kysytty tietyn aikajanan sisällä. |
| valvontaverkon ylimmän tason osakomponentti | Joku valvottavista kohteista ei vastaa tilatiedusteluun. |
| hälytyskanavan toimimattomuus | Häiriötilanteesta ilmoittaminen ylläpidolle epäonnistuu. |

Voidaan havaita, että ongelmat valvontajärjestelmässä kohdistuvat joko tiedon välitykseen verkon tasolta toiselle tai hälytyskanavan toimintaan. Muutoin kyseessä on järjestelmässä havaittu ongelma, josta ilmoitetaan pääkomponentin välityksellä ylläpidolle.

### 5.4.1 Alimman tason komponentin valvontatietojen hankinta epäonnistuu

Jos alimman tason osakomponentti ei suoriudu valvontaverkon sille asettamista velvoitteista tai sen toiminnassa on jotain tavallisuudesta poikkeavaa, havaitaan tämä ylemmän tason komponenteissa joko siten, että komponentti ei vastaa tilatiedusteluihin tai tämän lähettämissä tiedoissa on havaittavissa hälytysrajat rikkovia elementtejä. Ylimmän tason komponentti hälyttää tilanteessa ylläpidolle verkossa havaitusta ongelmasta.

### 5.4.2 Komponentin informaatiota ei olla tiedusteltu tietyn aikarajan puitteissa

Tällaisessa tilanteessa on oletettavaa ylemmän tason komponentin vioittuneen määrittelemättömästä syystä tai kyseessäolevan osakomponentin oma verkkoyhteys on vioittunut. Tämän ongelman havainneen komponentin on ensin varmistettava verkkoyhteyden toimivuus ylemmän tason osakomponenttiin ja jos tämä osoittautuu toimimattomaksi, aloitetaan valvontaverkon korjausoperaatio.

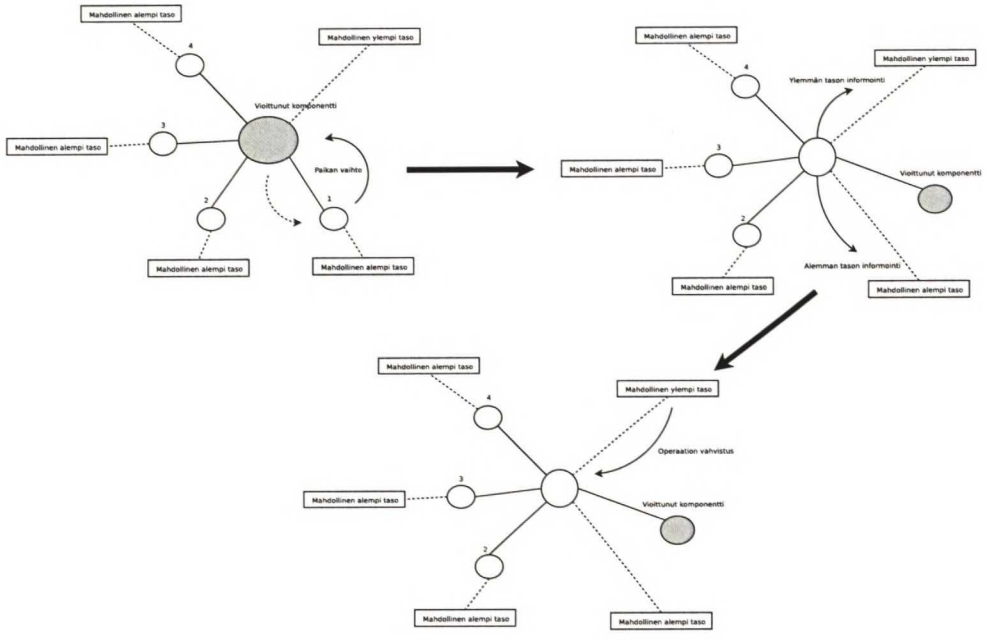
Toimimaton komponentti verkossa korvataan nostamalla yksi tämän vialisen komponentin alikomponenteista korvaamaan vioittunutta komponenttia ja tämä korvausoperaatio suoritetaan tarvittaessa jokaisella alemmallakin tasolla toimivuuden takaamiseksi. Täksi korvaavaksi komponentiksi valitaan alikomponenttien joukosta prioriteetiltaan korkein. Kun tämä vioittunut komponentti saadaan jälleen toimimaan, saa tämä paikan korvanneen komponentin alikomponenttina. Tämä prosessi on kuvattu tarkemmin luvussa 5.4.4.

### 5.4.3 Hälytysviestejä ei pystytä lähettämään ylläpidolle

Valvontajärjestelmän epäonnistuessa hälytysviestien lähetyksessä pääasiallista kanavaa käyttäen on pyrittävä käyttämään vaihtoehtoisia hälytystapoja ja jos nämäkään eivät auta tilanteessa, on tyydyttävä keräämään tietoa tietokantaan.

### 5.4.4 Menettely valvontaverkon ongelmatilanteessa

Valvontaverkon ongelmatilanteessa yksi tai useampi verkon osakomponentti ovat estyneitä suorittamaan osuuttaan verkkohierarkian edellyttämällä tavalla. Tähän syinä voivat olla verkkoyhteysongelmat, näihin komponentteihin kohdistuva mekaaninen ongelma tai ohjelmallinen virhetilanne. Koska valvontaverkosto ei häiriöistä huolimatta saa lamautua, vaan ongelmista on pyrittävä jokaisessa tilanteessa informoimaan ylläpitoa, on syytä muodostaa käytäntö osittaisen valvontaverkoston lamautumisen varalle. Tilanteessa, jossa tämä häiriö vaikuttaa yhtaikaisesti koko verkkoon, ei ymmärrettävästi korjausoperaatioista ole saavutettavissa haluttua hyötyä. Osittainen valvontaverkon ongelma voi kohdistua joko alimman tason, keskitason tai ylimmän tason verkkokomponenttiin. Koska valvontaverkko on määritelty rakenteeltaan hierarkiseksi, tulee vioittunut keski- tai ylemmän tason komponentti korvata nostamalla jokin komponentti alemmalta tasolta vioittuneen komponentin korvaajaksi. Alimman tason komponentin vioittuminen ei vaikuta valvontaverkon toimivuuteen, joten tämän osalta ei ole tarvetta suorittaa verkossa uudelleenjärjestelyjä.



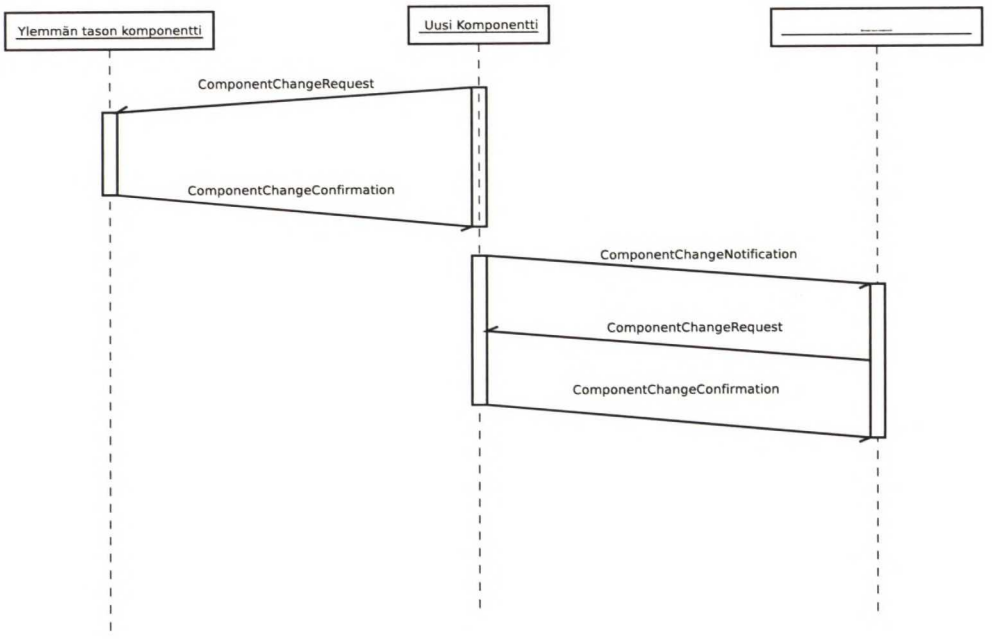

Kuva 5.5: Verkkokomponentin korvausprosessi

Vioittuneen komponentin korvaamiseksi tämän välittömät alikomponentit

etsivät keskuudestaan korkeimman prioriteetin omaavan jäsenen, joka korvaa vioittuneen komponentin valvontaverkossa. Tämä valittu komponentti ilmoittaa uudelle yläkomponentilleen komponenttimuutoksesta, jos kyseinen korvattu komponentti oli keskitason komponentti. Muutosprosessin jälkeen uusi vioittuneen komponentin toimintaa jatkava komponentti ilmoittaa entisille alikomponenteilleen verkkomuutoksesta ja tarvittaessa komponenttimuutosprosessi käydään läpi myös näillä alemmilla tasoilla rekursiivisesti.

Tilanteessa, jossa ongelmatilannetta korjattaessa joudutaan korvaamaan useampia verkkohierarkian tasoja ja verkkoyhteys muutoin on toimintakuntoinen, voi fyysinen verkon rakenne asettaa rajoituksia korvaamisprosessille. Tällaisessa tilanteessa korvaava komponentti tulkitaan ylimmän tason komponentiksi ja valvontaverkko pirstoutuu useampaan osaan. Näin valvontaverkolle saattaa muodostua useampia pääsolmuksi identifioituvaa komponenttia. Kukin näistä hajautuneen valvontaverkon pääsolmuista ryhtyy käsittelemään käytössään olevaa valvontainformaatiota ja ylläpidolle saadaan välitettyä näin tietoa osittaisverkkojen tilasta. Valvontaverkon keräämät tilatiedot pirstoutuvat tällaisessa verkon hajautumisessa pienempiin osakokonaisuuksiin ja verkon tilan seuraaminen on monimutkaisempaa, mutta akuutit hälytyssanomat saadaan välitettyä ylläpidolle.

Tätä verkkokomponentin korvausprosessia selventää kuva 5.5. Valvontaverkon komponenttien vaihtotapahtuma kulkee asiankäsittelydiagrammin 5.6 mukaisesti.




Kuva 5.6: Komponentin vaihdon asiankäsittelydiagrammi

Kun vioittuneet komponentit saadaan korjattua verkossa, pyytää nämä verkon ylimmän tason komponentilta oikeutta aiemman asemansa takaisin saamiseksi. Jotta palautuminen onnistuu järkevällä prosessoinnilla, verkko-

hierarkian palautus aloitetaan ylimmältä tasolta. Valvontaverkko pysyy aiemmassa häiriön jälkeisessä tilassa, kunnes ylimmän tason valvontakomponentti lähettää valvontaverkolle ilmoituksen verkon rakenteen muuttumisesta ja verkon palautuminen iteroidaan alemmissa verkon osissa vastaamaan nykyistä tilannetta. Oleellista tässä tilanteessa on, että nyt valvontaverkon ulkopuolelle joutuneiden komponenttien tilaa tiedustellaan ja toimivat komponentit kytketään jälleen osaksi valvontaverkkoa.

## 5.5 Tulevien ongelmatilanteiden ennustaminen

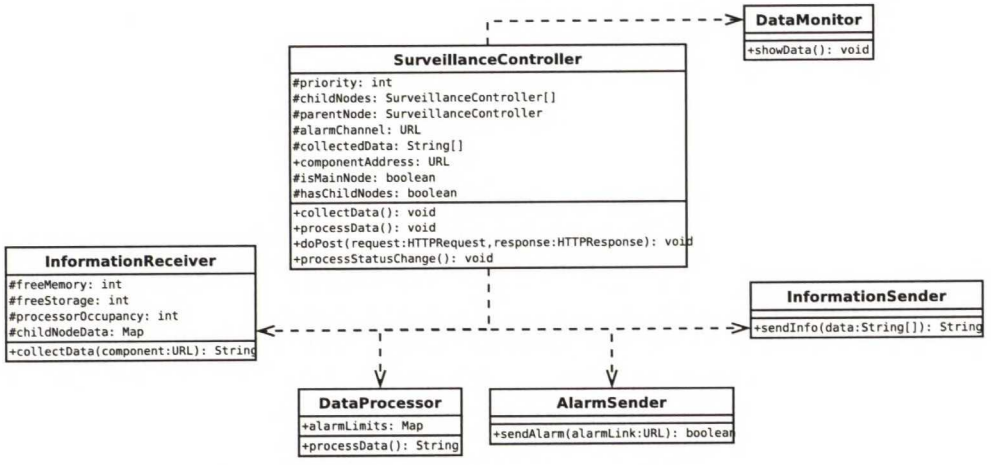
Hyvä valvontajärjestelmä pystyy tulkitsemaan valvontainformaatiota siten, että se pystyy reagoimaan tuleviin ongelmatilanteisiin jo ennakoivasti. Hajautetun ohjelmisto- ja laitealustan valvonnassa ennakoiminen kohdistuu pääosin muistin ja levytilan käyttäytymiseen, sillä verkko-ongelmia tai laitteiston rikkoutumista on hyvin hankalaa havaita valvontatiedoista ennen ongelman ilmenemistä. Levytilan ja muistin käytön tulevia ongelmatilanteita pystytään arvioimaan valvontatietojen muutoksien perusteella eli näiden resurssien loppuminen voidaan ennustaa suhteellisen tarkasti.

Valvontatiedon pitkäaikaisella seurannalla voidaan havaita myös järjestelmässä esiintyviä toistuvia ongelmatyyppejä, joita vastaan pystytään tiedon lisääntyessä kehittämään keinoja niiden estämiseksi. Tällaisesta esimerkkinä voivat olla toistuvat, säännölliset järjestelmän ylikuormittumistilanteet, jotka pystytään seurannan avulla rajaamaan tiettyihin valvontaverkon osakomponentteihin ja näissä suoritettaviin prosesseihin.

Tämän valvontatiedon hallinnoiminen pidemmältä aikaväliltä muuttuu haastelliseksi, jos valvontatietoa keräävä ja tulkitseva valvontaverkon ylimmän tason komponentti vaihtuu komponenttiongelmien vuoksi. Aiemmin kerätyt tiedot tulee pystyä synkronoimaan edelliseltä päätason komponentilta nykyiselle pääkomponentille, kunhan tämä vioittunut komponentti saadaan korjattua, mutta tällaisella menettelyllä joudutaan mahdollisesti siirtämään suuria tietomääriä tietoverkon komponentilta toiselle, joka voi mahdollisesti ruuhkauttaa verkon tiedonsiirron ja estää verkon varsinaisten palvelujen toimintaa. Tapa, jolla seurantatietojen siirtämistä komponentilta toiselle voidaan vähentää ongelmatilanteen jälkeen, on saattaa vioittunut valvontaverkon pääkomponentti toimintakuntoon ja nostaa tämä takaisin valvontaverkon pääkomponentiksi, jolloin tämän vioittunutta komponenttia korvanneen komponentin keräämät tiedot ovat oletettavasti suhteellisen vähäisiä ja näiden välittäminen pääkomponentille ei ruuhkauta tarpeettomasti käytettävää tietoverkkoa. Tätä seurantatietojen synkronointia ja siitä seuraavaa ennustettavuuden tarkentumista ei kuitenkaan tutkita tämän lopputyön puitteissa.

## 5.6 Ohjelman rakenne

Toteutettavan ohjelman on suoriuduttava tarvittaessa viidestä päätehtävästä: tiedonkeruusta, tiedon prosessoinnista, tietojen välittämisestä, hälyttämisestä ja annettava käyttäjälle käyttöliittymän valvontatietojen seurantaan. Tiedon keräämisessä tulee olla mahdollista hankkia niin kyseessäolevan tietojärjestelmän osan tilatiedot kuin tämän valvontaverkon hierarkian mukaisten alikomponenttien tilatiedot. Tiedon prosessointiin tulee olla mahdollisuus jokaisessa valvontaverkon osassa, mutta jos komponentti ei ole ylimmän tason komponentti, prosessointia ei määrittelyjen mukaan suoriteta. Tietojen välittämisellä tarkoitetaan, että jokaisen valvontaverkon osakomponentilla tulee olla valmius välittää keräämänsä valvontatieto ylemmän tason valvontakomponentille. Hälyttämiseen tulee jokaisella komponentilla olla valmius, vaikka hälytykset lähetetään määritelmän mukaisesti ylimmän tason komponentilta. Ohjelmalla tulee olla valmius esittää sen aliverkon valvontatiedot, joiden valvontavastuu sillä on verkkohierarkiassa. Tämä ohjelman rakenne on kuvattu UML-kaaviona kuvassa 5.7.




Kuva 5.7: Valvontaohjelman UML-kaavio

## 5.7 Tiedonvälitysrajapinta

Valvontatietojen välittämiseksi komponentilta toiselle tulee käyttää ytimekästä ja silti kuvaavaa tiedonvälitystapaa, koska komponenttien välillä ei ole jatkuvaa tiedonsiirtoa. Tiedonvälitykseen käytetään XML-sanomapohjaista viestiä, jolloin sanoman sisällön rakenteen määrittämisellä pystytään spesifioimaan sanoman lähettäjä ja välitettävät tiedot pystytään ilman tulkinnan varaa lajittelemaan oikeille komponenteille. XML on metakieli, jonka avulla voidaan kehittää rakenteisia kuvauskieliä ja näitä voidaan käyttää niin dokumentin rakenteen kuvaamiseen kuin sovellusten väliseen tiedonsiirtoon, kuten Ruini esittelee oppaassaan [14]. Valittu sanomatyyppi antaa mahdollisuuden välittää useampien komponenttien tietoja yhden välitettävän viestin si-

sällä yhtäaikaa ja viesti voidaan lähettää normaalin TCP/IP yhteyden välityksellä.

Suunniteltuja viestityyppejä, joita on esitetty aeimmin kappaleissa 5.1.1 ja 5.4.4, ovat määritelmien mukaisesti tietojen pyyntisanoma, tietojen välityssanoma, tietojen vastaanottamisen kuittaussanoma, konfiguraation muutospyyntösanoma, konfiguraation muutosilmoitussanoma ja konfiguraation muutoksen vahvistussanoma. Näiden avulla pystytään hakemaan valvottavilta verkon komponenteilta tilatietoja ja ongelmatilanteiden ilmaantuessa kyetään verkko saattamaan valvontatietoja välittävään tilaan. Näiden sisältämä informaatio esitellään seuraavaksi.

### 5.7.1 Tietojen pyyntisanoma

Tämän sanoman tarkoituksena on välittää heräte komponentin alikomponentille, että herätteen antanut ylemmän tason komponentti tarvitsee tämän pyynnön kohteena olevan komponentin ja sen valvomien alikomponenttien valvontatietoja. Sanoma on muodoltaan seuraavanlainen:

**Algorithm 4** Tietojen pyyntisanoma

```
<StatusRequest>
 <requestID>requestID<requestID>
 <requestedComponent>targetComponentID
 </requestedComponent>
<requestingComponent>requestingComponentID
 </requestingComponent>
</StatusRequest>
```

Sanoman requestID-elementti yksilöi tapahtuman herätteen antaneelle komponentille, sillä sama tunniste välitetään myös vastaussanomassa.

Sanoman requestedComponent-elementti osoittaa herätesanoman tietylle alikomponentille ja sanoman requestingComponent-elementti identifioi valvontatietoja pyytäneen komponentin.

### 5.7.2 Tietojen välityssanoma

Tämä sanoma välittää komponentin ja tämän alikomponenttien uusimmat valvontatiedot tietoja pyytäneelle ylemmän tason komponentille. Sanoma on muodoltaan seuraavanlainen:

**Algorithm 5** Tietojen välityssanoma

```
<StatusInfo>
<requestID>requestID<requestID>
<requestingComponent>requestingComponentID
</requestingComponent>
<components>
 <component id="1">
  <status>
<timestamp>20050308100000</timestamp>
<value>OK</value>
</status>
  <diskSpace>
<timestamp>20050308100000</timestamp>
<value>1500000</value>
</diskSpace>
  <memory>
<timestamp>20050308100000</timestamp>
<value>50000</value>
</memory> <load>
<timestamp>20050308100000</timestamp>
<value>0.8</value>
</load>
 </component>
 <component id="2">
  <status>
<timestamp>20050308100000</timestamp>
<value>OK</value>
</status> <diskSpace>
<timestamp>20050308100000</timestamp>
<value>23000000</value>
</diskSpace>
  <memory>
<timestamp>20050308100000</timestamp>
<value>250000</value>
</memory>
  <load>
<timestamp>20050308100000</timestamp>
<value>0.3</value></load>
 </component>
 <component id="3">
  <status>
<timestamp>20050308100000</timestamp>
<value>NOK</value>
</status>
 </component>
</components>
</StatusInfo>
```

Sanoman requestID- ja requestingComponent-elementit identifioivat mille ylemmän tason komponentille tilatiedot välitetään ja tämä ylemmän tason komponentti pystyy yhdistämään tämän vastausviestin välittämäänsä herätesanomaan.

Sanoman oleellinen osio on components-elementti, joka sisältää kaiken komponentin keräämän valvontatiedon valvottavalta verkon osaltaan komponenteittain component-elementeissä. Kukin component-elementti sisältää attribuuttina komponentin tunnisteen, komponentin tilan status-elementissä ja muut valvottavat tilatiedot kuten vapaan käyttömuistin määrän erillisissä elementeissään. Kukin erillinen tilatieto sisältää myös aikaleiman, jolloin tämä viimeisin tieto on järjestelmästä otettu.

### 5.7.3 Tietojen vastaanottamisen kuittaussanoma

Tämän sanoman tarkoituksena on vahvistaa alikomponentille, jolta edellä esitellyin viestein on tiedusteltu valvontatietoja, että tämän välittämät tiedot on käsitelty. Sanoma on muodoltaan seuraavanlainen:

**Algorithm 6** Tietojen vastaanottamisen kuittaussanoma

```
<StatusConfirmation>
<requestID>requestID<requestID>
<status>OK</status>
<requestedComponent>targetComponentID
</requestedComponent>
<requestingComponent>requestingComponentID
</requestingComponent>
</StatusConfirmation>
```

Sanoman requestID-elementti identifioi sanoman kuuluvaksi viestiketjuun, jonka ylemmän tason komponentti aloitti tietojen pyyntisanomalla. Sanoman requestedComponent-elementti identifioi komponentin, jolta tilatiedot on vastaanotettu ja requestingComponent-elementti puolestaan komponentin, joka on vastaanottanut tiedot. Sanoman status-elementti ilmaisee, onko tilatiedot pystytty käsittelemään onnistuneesti.

### 5.7.4 Konfiguraation muutospyyntösanoma

Tämän sanoman tarkoituksena on pyytää verkon korkeamman tason komponentilta lupaa korvata jokin verkon komponenteista toisella. Kyseessä voi olla joko vikatilanteen havaitsemisen jälkeinen valvontaverkon paikkausoperaatio tai korjatun tilanteen jälkeinen palautumisoperaatio. Sanoma on muodoltaan seuraavanlainen:

**Algorithm 7** Konfiguraation muutospyyntösanoma

```
<ComponentChangeRequest>
 <requestID>requestID</requestID>
 <upperComponent>componentID
</upperComponent>
 <replacingComponent>componentID
</replacingComponent>
 <componentToBeReplaced>componentID
</componentToBeReplaced>
</ComponentChangeRequest>
```

Sanoman requestID-elementti identifioi aloitettavan sanomanvälitystapahtuman. Tätä tunnistetta käytetään ylemmän tason välittämässä paluusanomassa.

Sanoman upperComponent-elementti osoittaa, minkä komponentin alikomponentille muutosta ollaan tekemässä, replacingComponent-elementti osoittaa komponentin, joka tulee korvaamaan osoitetun komponentin ja componentToBeReplaced-elementti kertoo, mikä komponentti halutaan korvata.

### 5.7.5 Konfiguraation muutoksen vahvistussanoma

Tämän sanoman tarkoituksena on joko vahvistaa tai evätä valvontaverkostoon kohdistuva muutos. Jos muutos on hyväksyttävissä, esitetään sanoman ohessa korvattavan verkkokomponentin alikomponentit hierarkisesti. Sanoma on muodoltaan seuraavanlainen:

**Algorithm 8** Konfiguraation muutoksen vahvistussanoma

```
<ComponentChangeConfirmation>
 <requestID>requestID</requestID>
 <updatedComponent>componentID
</updatedComponent>
 <previousComponent>componentID
</previousComponent>
 <status>OK</status>
 <components>
  <component level="1">componentID</component>
  <component level="1">componentID</component>
  <component level="1">componentID</component>
  <component level="2" upper="componentID">
componentID</component>
 </components>
</ComponentChangeConfirmation>
```

Sanoman requestID-elementti identifioi sanomanvälitystapahtuman, jonka alikomponentti on aiemmin aloittanut.

Sanoman updatedComponent-elementti osoittaa, mikä komponentti saa uuden statuksen valvontaverkossa. Sanoman previousComponent-elementti il-

maisee, mikä valvontaverkon komponentti tullaan korvaamaan. Sanoman status-elementti ilmaisee, hyväksytäänkö tehtävää muutosta. Sanoman components-elementti sisältää tämän korvattavana olevan komponentin aliverkon rakenteen siten, että kukin component-elementti sisältää aiemman aliverkon komponentin tunnisteen ja attribuutteina komponentin tason aiempaan yläkomponenttiin verrattuna ja mahdollisen yläsolmun tunnistenumeron.

### 5.7.6 Konfiguraation muutosilmoitussanoma

Tämän sanoman tarkoituksena on ilmoittaa valvontaverkostoon kohdistuvasta muutoksesta ja tapahtuvista verkon muutoksista. Sanoma on muodoltaan seuraavanlainen:

**Algorithm 9** Konfiguraation muutosilmoitussanoma

```
<ComponentChangeNotification>
 <upperComponent>componentID</upperComponent>
 <directLowerComponents>
  <component>componentID</component>
  <component>componentID</component>
  <component>componentID</component>
 </directLowerComponents>
 <otherComponents>
  <component upper="componentID">
componentID</component>
  <component upper="componentID">
componentID</component>
  <component upper="componentID">
componentID</component>
 </otherComponents>
</ComponentChangeNotification>
```

Sanoman upperComponent-elementti osoittaa, minkä komponentin alikomponentille tullaan tekemään muutosta.

Sanoman directLowerComponents-elementti sisältää ne komponentit, jotka tulevat olemaan muutoksen jälkeen tämän komponentin alikomponentteja.

Sanoman otherComponents-elementti sisältää loput komponentin alikomponentit, jotka eivät ole suorassa yhteydessä tähän komponenttiin, mutta kuitenkin jollain tavalla tämän alikomponentteihin. component-elementin upper-attribuutti identifioi komponentin, jonka alikomponentiksi tämä kyseinen solmu kuuluu.

## 5.8 Tiedonvälityksen turvaaminen

Hajautetun palvelinympäristön osakomponentit voivat sijaita fyysisesti hyvin erillään toisistaan, joka aiheuttaa sen, että tiedonsiirtämisessä komponentilta toiselle joudutaan väistämättä käyttämään myös julkista Internet-verkkoa eli

tiedonvälitystä komponentilta toiselle ei pystytä kaikissa tapauksissa toteuttamaan käyttäen ainoastaan yrityksen sisäverkkoa tiedonsiirtokanavana. Aiemmin luvussa 5.7 esitetty tietojen välitystapa käyttäen XML-sanomia soveltuu käyttöön käytettävästä välitysverkosta riippumatta, mutta tiedonvälittämisessä julkisessa verkossa riski tietojen joutumisesta kolmannen osapuolen käsiin kasvaa. Tästä johtuen on syytä harkita vastatoimia, jotta väärille tahoille joutuneet viestit ovat heille teknisesti niin vaikeita tulkita, että kustannuksellisesti saavutettava hyöty viestien onnistuneesta tulkinnasta jää minimaaliselle tai olemattomalle tasolle. Sovelias tapa XML-sanomien tietosisällön väärinkäytön estämiseksi on muuntaa tietosisältö salausalgoritmeja hyödyntäen ulkopuoliselle ymmärtämättömään muotoon.

Salausmenetelmän valinnan kannalta tulee valita salausmenetelmäksi riittävän kevyt toteutus, jottei salauksen käsittely kuormita yksittäistä komponenttia liialti, mutta silti riittävän tehokas ehkäisemään tietovuotoja kolmannelle osapuolelle. Viestintävirasto määrittelee tietoturvaa käsittelevässä julkaisussaan [21], että symmetrinen salaus on erittäin nopea salausmuoto, mutta ongelmana on salausavaimen turvallinen jakaminen eri osapuolten kesken. Koska salatun sanoman lähettäjä ja vastaanottaja on valvontasovelluskokonaisuudessa saman tahon hallinnoima ja salauksen hallinnassa painotetaan salausmenetelmän nopeutta, on symmetrisen salauksen käyttäminen, jossa sekä salaus että purkaminen suoritetaan käyttäen samaa salausavainta, perusteltua.

## 5.9 Yhteenveto

Automaattisen valvontasovelluksen toteuttamiseksi tässä luvussa määriteltiin olennaisia ominaisuuksia toimivuuden mahdollistamiseksi. Viestiliikenteen minimoimiseksi valvontatietoja välitetään ainoastaan alemman tason komponentilta tämän yläkomponentille tähtirakenteen mukaisesti ja näin aina korkeimmalle verkon tasolle asti, jossa valvontatiedot prosessoidaan ja reagoidaan mahdolliseen ongelmatilanteeseen.

Valvontatiedot välitetään xml-sanomilla vasta, kun ylemmän verkkotason komponentti pyytää komponentilta tämän keräämiä valvontatietoja. Valvontatiedot tallennetaan tietokantaan ja jotta tietomäärä ei kasva hallitsemattomasti, mutta historiatietoja järjestelmän käyttäytymisestä silti säilyy, tietokannan vanhempia kerättyjä tietoja harvennetaan.

Koska järjestelmä, jota valvotaan, voi hajautua useaan maantieteelliseen kohteeseen, on valvontatiedot lähes poikkeuksetta lähetettävä ainakin osittain julkista Internet-verkkoa hyödyntäen. Lähetettävät valvontatietoja sisältävät sanomat salataan symmetristä salausta hyödyntäen, sillä valvontaverkko on saman tahon alaisuudessa ja salausmetodin tulee olla nopea, jottei tämä vie liikaa tuotantokäyttöön suunniteltujen tietojärjestelmien resursseja.

# Luku 6

# Automaattisen valvontasovelluksen toteutus

Luvussa summataan valvontasovelluksen toteutuksen yhteydessä tehtyjä rajauksia ja yleisiä havaintoja.

Automaattisen valvontaohjelman toteuttaminen on riippuvainen valvottavan tietojärjestelmän rakenteesta ja ominaisuuksista. Tietojärjestelmään, joka on kuvattu luvussa 4.4, oli valvontaohjelman tiedonsiirron toteuttamisessa hyödynnettävissä käytettävän ohjelmistoalustan toteutusta. Tietojärjestelmän jokaiselle laitealustalle on asennettuna Jboss J2EE-sovelluspalvelin, kaikilla testiympäristön laitealustoilla on Linux-ytimeen perustuva käyttöjärjestelmä ja jokaiselle on asennettuna PostGreSQL-tietokanta. Nämä testiympäristön ohjelmistovalinnat omalta osaltaan ohjasivat toteutuksessa tehtyjä valintoja.

## 6.1 Ohjelmointikielen päättäminen

Valvontasovelluksen idean mukaisesti sitä on pystyttävä suorittamaan kullakin tietojärjestelmän erillisellä komponentilla ja sen on pystyttävä tukemaan jokaisen komponentin ominaispiirteitä. Jokainen tietojärjestelmän komponentti voi omata N kappaletta erilaisia valvottavia suureita, kuten käyttömuistin tila, massamuistin käyttöaste, prosessorin kuormitus ja niin edelleen. Valvontaohjelman konfigurointi komponenttikohtaisesti on näinollen sangen ratkaisevassa asemassa joustavan valvontasovelluksen toteutuksen kannalta, jotta valvottavasta tietojärjestelmästä saataisiin kerättyä niitä valvontatietoja, jotka ovat ylläpidollisesti merkittävässä asemassa.

Valvontasovelluksen toteutuksen kannalta on tietoliikenneverkon hyödyntämisen oltava mahdollista siten, että jokainen tietojärjestelmän komponentti, joka on valvontasovellukseen kytkettynä, kykenee lähettämään ja vastaa-

nottamaan valvontatietoja tai valvontaverkon konfiguraatiomuutoksia muun toiminnan tästä häiriintymättä.

### 6.1.1 Java

Päätös valvontasovelluksen ytimen ohjelmoimiseksi Javalla oli sangen suoraviivainen prosessi, sillä olihan jokaiselle testiympäristön laitealustalle asennettuna JBoss J2EE-sovelluspalvelimet, joiden avulla tietoliikenteen organisointi komponenttien välillä oli joustavaa ja hallittua. Javan etuja valinnan kannalta muihin ohjelmointikieliin verrattuna, kuten C tai C++, oli myös laitealustariippumattomuus, jolloin samaa käännettyä sovellusta voidaan suorittaa millä tahansa laitealustalla, jolle Java-ympäristö on asennettuna, konfiguraatiotiedoston muokkaamisen jälkeen. Oliopohjaisena kielenä Javalla pystyi myös erittelemään valvontasovelluksen erilliset toiminnallisuudet modulaarisiin osakokonaisuuksiin, kuten luvussa 5.6 esitettiin.

Vaikka Java on Web-tekniikoissa etenkin Client-Server-toteutuksissa paljon käytetty, on sen rajoitteena virtuaalinen ajoympäristö, joka tekee laitealustan valvonnan haasteelliseksi tässä valvontasovelluksen tapauksessa. Javan ominaisuuksiin kuuluu kuitenkin mahdollisuus ajaa virtuaaliympäristön ulkopuolisia ohjelmia ja vastaanottaa näiden palauttamaa tietoa. Tällöin itse ongelmasta tuleekin vahvuus, sillä tällä tavalla jokaisen komponentin tavallisesta konfiguraatiosta poikkeavat valvottavat osa-alueet voidaan toteuttaa itse valvontasovelluksen ulkopuolisella ohjelmalla ja tiedot voidaan liittää osaksi valvontasovellusta vähäisin konfiguraatioin.

### 6.1.2 Komentotulkkiskriptit sovelluksen osana

Linuxin komentotulkille voidaan kirjoittaa skriptejä, joiden avulla voidaan kerätä tietoja useasta eri lähteestä yhteen ja näiden sisältöä voidaan muokata haluttuun muotoon. Valvontasovelluksen kannalta tällaisien skriptien avulla voidaan koostaa laitealustasta kerättyä valvontatietoa valvontasovelluksen käyttöön ja näin suoriutua mitä erinäisempien valvontakohteiden valvonnasta.

Kuten Smith ja Henry ovat havainneet tutkiessaan Javan soveltumista klusterien valvontaan tämän suorituskyvyn kannalta [15], on Linuxin virtuaalinen /proc-tiedostojärjestelmä sangen tehokas kohde kerätä laitteiston tilaa indikoivaa tietoa. /proc-tiedostojärjestelmä luotiin alunperin erillisten prosessien tietojen yksinkertaiseen tutkintaan UNIX-, Solaris-, Linux- ja BSD-käyttöjärjestelmille, mutta Linuxin tapauksessa tätä käytetään nykyisin kaikkeen Linux-ytimen raportoivan tiedon keräämiseen. Toisin kuin Smith ja Henry, jotka päätyivät lukemaan tutkimuksessaan /proc-tiedostojärjestelmän tietoja suoraan Javalla, päädyin ratkaisuun prosessoida /proc-tiedostojärjestelmän tietoja ensin komentotulkkiskriptein soveltuvampaan muotoon. Näin erilaisien parametrien lisäys valvontasovellukseen ei luo muutospaineita itse valvontasovellukseen, vaan muutokset voidaan tehdä komentotulkkiskripteihin

ja lisätä tämä valvontasovelluksen konfiguraatioihin.

Valvontatietojen kerääminen tietojärjestelmän komponentilta Javan ulkopuolisin skriptein tai ohjelmin antaa mahdollisuuden sovittaa valvontasovelluksen hyvin monenlaiselle laitealustalle, kunhan tähän on asennettuna Java-ympäristö ja jokin sovelluspalvelin tiedonsiirron mahdollistamiseksi komponenttien välillä.

## 6.2 Valvontaverkon sanomaliikenteen salaus

Kuten jo aiemmin luvussa 5.8 todettiin, on valvontasovelluksen osakomponenttien välinen tiedonsiirto syytä salata, sillä valvontatietoja joudutaan usein kuljettamaan julkisessa Internet-verkossa. Toteutuksessa salausalgoritmiksi otettiin symmetrinen salausalgoritmi 3DES, jossa kolmella 64-bittisellä salausavaimella kryptataan salattava data siten, että aina edeltävän kryptauksen jälkeinen tulos kryptataan seuraavalla avaimella DES-algoritmin mukaisesti. Tämä salausalgoritmi on sangen nopea, jolloin valittu ratkaisu ei myöskään kuluta tarpeettomasti tietojärjestelmän komponentin resursseja.

Ohjelman kannalta oli tarpeellista salata ainoastaan lähetettävien xml-sanomien komponenttispesifiset osat, jolloin myös sanomien salattava osuus jää pienemmäksi ja viestien käsittely tehostuu entisestään tarvittavan tietoturvan silti täyttyessä.

## 6.3 Ohjelmistototeutukseen tehdyt rajaukset

Valvontasovelluksen toteuttaminen määrittelyn laajuisena kokonaisuutena olisi ollut lopputyön oman työn osuutena suhteettoman laaja, joten muutamia osakokonaisuuksia on toteutuksessa yksinkertaistettu tai jätetty käsiteltäväksi sovelluksen jatkokehityksen yhteydessä. Rajaukset kohdistuivat pääosin tietokantaan, käyttöliittymään ja verkon manipulointiin vikatilanteesta selviytymisen jälkeen.

### 6.3.1 Tietokanta

Tietokannan määrittelyssä luvussa 5.3 esiteltiin tietokannan sisällön asteittainen poistaminen, jottei tietokannan koko kasva ajan myötä tarpeettoman raskaaksi historiatietojen käyttötarkoitukseen nähden. Tämä tietokannan sisällön karsiminen jätettiin kuitenkin toteutuksesta pois, sillä tämä ei kuitenkaan vaikuta itse valvontatietojen keräämiseen ja tulkitsemiseen, johon tutkinta valvontasovelluksen osalta pääosin kohdistuu.

### 6.3.2 Käyttöliittymä

Valvontasovelluksen käyttöliittymä on toteutettu ainoastaan sovelluksen toiminnallisuuksien testaamisen näkökulmasta, jonka vuoksi graafisia elementtejä ei ole käyttöliittymään toteutettu. Käyttöliittymä päivittää itse itsensä

minuutin välein siten, että näyttää kaikkien valvottavien komponenttien valvottavien suureiden tämän hetkiset arvot taulukossa ja erillisessä listauksessa ilmoittaa valvontasovelluksen kymmenen viimeksi havaitsemaa häiriötä. Ajan käyttäminen valvontatietojen esteettisempään esittämiseen sovelluksen tutkimisvaiheessa ei ole perusteltua.

### 6.3.3 Vikatilanteesta palautuminen normaalitilaan

Valvontasovelluksen määrittelyiden mukaisesti valvontasovellus pyrkii aktiivisesti välittämään jokaisen valvottavan verkon osakomponentin tiedot hierarkiassa ylemmälle tasolle. Tämä tarkoittaa, että verkossa alemman tason komponentti reagoi pyytämällä verkkohierarkian muutosta, jos tämän suora ylemmän tason komponentti ei tiedustele tilatietoja konfiguroidun aikarajan puitteissa.

Määritelmän mukaan verkko kykenee myös palautumaan tästä vikatilasta heti, kun verkon viallinen komponentti palautuu normaaliksi. Tämä ominaisuus on kuitenkin jätetty toteuttamatta testikonfiguraatiossa aikataulullisista syistä, sillä hierarkiseen verkkoon on saatettu vikatilanteen esiintyessä suorittaa useita hierarkiamuutoksia ja näiden palauttaminen korjatun komponentin palautuessa takaisin valvontaverkkoon osoittautui haastavaksi ongelmaksi.

### 6.3.4 Ongelmatilanteiden ennustaminen

Valvontasovelluksen historiatietojen keräyksen pääasiallisena tarkoituksena oli luvun 5.5 mukaisesti havaita järjestelmän toistuvia käyttäytymismalleja ja arvioida mahdollisia seuraavia ongelman esiintymisaikoja. Tämä problematiikka on kuitenkin laajuudessaan jo täysin oman tutkimusaiheensa arvoinen ja jätetään yhdeksi valvontasovelluksen jatkokehitysmahdollisuudeksi.

## 6.4 Määritelmiin tehdyt muutokset

Valvontasovelluksen määritelmiin toteutusvaiheessa tehdyt muutokset kohdistuivat tiedonvälityksen XML-sanomiin.

Tilatietojen välitykselle määritetyt vahvistussanomat, StatusConfirmation, havaittiin tarpeettomiksi, sillä epäonnistunut tiedonvälitys havaittaisiin myös tiedonvälityksen aikakatkaisuna. Sanoman poistamisella vähennetään myös tarpeetonta verkon kuormitusta.

Verkon tilan muutoksen vahvistussanoma, ComponentChangeConfirmation, koettiin myös edellä mainitun syyn mukaisesti tarpeettomaksi ainakin pienimuotoisessa testiympäristössä, sillä komponentille sallitut verkkomuutokset tarkistetaan niin alemmalla tasolla kuin ylemmällä, muutoksen hyväksyvällä, tasolla. Laajemmassa valvontaverkossa vahvistussanoman käyttäminen on kuitenkin perusteltua.

Verkon muutoksesta kertovan ComponentChangeNotification sanoman rakennetta muokattiin toteutusvaiheessa siten, että sanoma sisältää koko verkon hierarkian. Sanoma on muutetussa muodossaan seuraavanlainen:

**Algorithm 10** Konfiguraation muokattu muutosilmoitussanoma

```
<ComponentChangeNotification>
<components>
 <component upper="componentID">componentID</component>
<component upper="componentID">componentID</component>
<component upper="componentID">componentID</component>
<component upper="componentID">componentID</component>
 </components>
</ComponentChangeNotification>
```

Konfiguraation muutossanomaa on näinollen muokattu siten, että koko verkon hierarkia välitetään jokaiselle verkon osakomponentille.
component-elementin upper-attribuutti ilmaisee komponentin välittömän yläkomponentin.

## 6.5 Yhteenveto

Valvontasovelluksen toteutuksessa päädyttiin käyttämään Java-ohjelmointikieltä ohjelman logiikan ja tiedonsiirron toteuttamiseen. Valvottavat tiedot tietojärjestelmän osakomponentilta kerätään hyödyntäen komentotulkkiskriptejä, jolloin valvontasovellus kyetään konfiguroimaan useille erilaisille laitealustoille soveltuvaksi vain näitä komentotulkkiskriptejä kehittämällä.

Sovelluksen toteutuksessa tietokannan tietomäärän asteittainen harventaminen, käyttöliittymän ulkoasun kehittäminen, ongelmatilanteiden ennustaminen ja vikatilanteista palautuminen normaaliin valvontatilaan koettiin laajentavan toteutettavaa sovellusta huomattavasti, jolloin myös tämän toiminnallisuuksien testaaminen olisi vaikeutunut. Näinollen nämä ominaisuudet päätettiin siirtää valvontasovelluksen tulevaisuuden kehityssuunnitelmiin.

Valvontasovelluksen sanomaliikennettä yksinkertaistettiin toteutuksen yhteydessä, sillä etenkin valvontaverkon valvontatiedon vahvistussanoma havaittiin sangen vähäpätöiseksi informaatioarvoltaan.

# Luku 7

# Testaus

Tämän luvun tarkoituksena on käsitellä niitä menettelytapoja, joiden avulla kehityksen kohteena olevan valvontasovelluksen toimivuutta käytännössä pyrittiin todentamaan ja mahdolliset ongelma-alueet rajaamaan.

Ohjelmiston testaus voidaan jakaa eri tasoihin niin sanotun V-mallin mukaisesti, jonka Haikala ja Märijärvi määrittävät koostuvaksi moduulitestauksesta, integrointitestauksesta ja järjestelmätestauksesta [5]. Moduulitestauksessa ideana on keskittyä yksittäiseen itsenäiseen ohjelman osaan eli moduuliin, jonka toimivuutta verrataan testaamalla teknisen määrittelyn asettamiin vaatimuksiin. Integraatiotestauksessa yhdistetään useampia moduuleja ja keskitytään näiden välisten rajapintojen toimivuuden varmistamiseen. Järjestelmätestauksessa puolestaan keskitytään vertaamaan ohjelmistoa kokonaisuudessaan siten, että saatuja testituloksia verrataan ohjelmiston määrittelydokumentaatiossa asetettuihin vaatimuksiin.

Valvontasovelluksen testaaminen keskittyy edellä mainittujen kolmen testaustason kahteen ensimmäiseen tasoon eli moduulitestaukseen ja integrointitestaukseen pääosin luvussa 6.3 esiteltyjen rajausten vuoksi, joiden johdosta kaikkia valvontasovelluksen määrittelyssä esitettyjä valvontasovelluksen ominaisuuksia ei tämän lopputyön puitteissa voida testata järjestelmätestauksen tasolla.

## 7.1 Moduulitestaus

Jaettaessa valvontasovelluksen ohjelmistoa itsenäisiin kokonaisuuksiin, jotka suorittavat tehtäväänsä toisista osista riippumatta, voidaan näiksi luokitella valvontatiedon lukeminen yksittäiseltä komponentilta,valvontatiedon pyytäminen toiselta komponentilta, valvontatiedon lähettäminen toiselle komponentille, tilatietojen esittäminen käyttöliittymässä, tietojen salaus ja salauksen purkaminen.

### 7.1.1 Valvontatiedon lukeminen laitealustalta

Valvontatiedon lukemisen varmistaminen laitealustalta sovelluksen tietokantaan testattiin käynnistämällä pelkästään valvontaa suorittava ohjelman säie, konfiguroimalla tämä lukemaan komponentin tilatiedot ja tallentamaan nämä valvontasovelluksen tietokantaan.

Itse tilatietojen lukemiseen laitealustalta käytettiin erillisiä komentotulkkiskriptejä, joiden avulla valvontaa suorittava valvontaohjelmiston säie vastaanotti valvottavien suureiden tämän hetkiset arvot ja päivitti nämä onnistuneesti valvontasovelluksen tietokantaan.

### 7.1.2 Valvontatiedon pyytäminen toiselta komponentilta

Valvontatiedon pyytäminen verkkohierarkiassa komponentin alikomponenteilta testattiin käynnistämällä pelkästään valvontapyynnöt lähettävä säie ja konfiguroimalla tämä säie siten, että tämä pystyi tietokannasta lukemaan valvottavien komponenttien verkko-osoitteet ja muodostamaan komponentin tilatietokyselysanoman. Muodostettu sanoma kirjoitettiin tiedostoon niin sanomaa muodostettaessa kuin tämän vastaanottoa simuloivassa ohjelman osassa ja näiden sisällön oikeellisuus määrityksiin verrattuna varmistettiin.

### 7.1.3 Valvontatiedon lähettäminen toiselle komponentille

Valvontatiedon lähettäminen toiselle valvottavan verkon komponentille testattiin luomalla valvontatietosanoman koostava moduuli ja alustaen tämän oletettu valvottava komponentti tietokannasta löytyväksi. Suorituksen jälkeen muodostunutta valvontatiedot sisältävää xml-sanomaa verrattiin tietokannan arvoihin, jotta näiden oikeellisuus voitiin todeta.

### 7.1.4 Sanomaliikenteen salaus

Valvontatiedon sanomaliikenteen salauksen toimivuuden testaamiseksi luotiin valvontasovelluksessa käytettäviä xml-sanomia ja tallennettiin nämä tiedostoon ennen ja jälkeen salaustoimenpiteen.

### 7.1.5 Sanomaliikenteen salauksen purku

Valvontatiedon sanomaliikenteen salauksen purkaminen todennettiin antamalla salauksen testaamisen tuotteena saadut tallenteet syötteenä salauksen purkamista suorittavalle ohjelmalle. Saatuja tuloksia verrattiin salauksen kohteina olleisiin xml-sanomiin.

### 7.1.6 Komponentin tiedon esitys käyttöliittymässä

Valvontatiedon esittäminen käyttöliittymässä todennettiin vertaamalla tietokannasta saatuja arvoja käyttöliittymän esittämiin tietoihin.

## 7.2 Integraatiotestaus

Valvontasovelluksen ohjelmiston integraatiotestaamisen osuus rajapintojen toimivuuden testaamiseksi rajoittuu sanomaliikenteen vastaanotto-operaatioiden varmentamiseen.

### 7.2.1 Tiedon vastaanottaminen toiselta komponentilta

Valvontasovelluksen toiminta perustuu xml-sanomaliikenteellä käytävään kommunikointiin. Kommunikoinnin tarkoituksena on välittää valvottavan ympäristön valvontatietoja toisille verkon komponenteille tai tiedottaa valvontaverkossa havaitusta ongelmakohdasta ja tämän mahdollisesta korjaustoimenpiteestä.

Tiedon vastaanottamista testattiin pystyttämällä kaksi erillistä laitealustaa, joista toinen konfiguroitiin edustamaan valvontaverkon ylimmän tason komponenttia ja toinen tämän alikomponenttia. Ylemmän tason komponentti määritettiin pyytämään alemman tason komponentilta komponentin tilatietoja ja vastaanotettujen tilatietojen todentaminen suoritettiin vertaamalla tietokantojen tietoja keskenään.

## 7.3 Järjestelmätestaus

Järjestelmätestauksessa pyritään todentamaan ohjelmistolle määrittelyvaiheessa asetettujen määrittelyjen realisoituminen. Valvontasovelluksen toimivuuden kannalta on tärkeää varmentaa, että sovellus on kykeneväinen näyttämään valvottavan verkon komponenttien valvottavien suureiden kulloisenkin tilan ja havaitsee mahdolliset ongelmatilanteet niin valvottavien suureiden, valvottavien tai valvovien komponenttien osalta. Myös valvontasovelluksen vikatilanteessa käyttäytymisen varmentaminen on tärkeää.

### 7.3.1 Valvottavan järjestelmän tilatietojen esittäminen

Valvontasovelluksen tilatietojen esittäminen käyttöliittymässä todennettiin moduulitestauksen kaltaisesti, joskin useamman komponentin tiedot esitettiin samanaikaisesti. Testisovelluksen valvontatietojen päivityksen yhteydessä myös käyttöliittymän tietojen tuli päivittyä jokaisen komponentin osalta.

### 7.3.2 Vikatilanteen havaitseminen järjestelmän tilatiedoista

Valvontasovelluksen tilatietojen seurannassa on jokaisella eri kategoriaan kuuluvalla suureella erilaiset kriteerit siitä milloin luettu arvo on normaaliksi, milloin epänormaaliksi luokiteltavissa. Valvontasovelluksen tulee havaita muun muassa jos vapaan käyttömuistin määrä vähenee kriittisesti tai jos prosessorin käyttöaste nousee riittävän korkeaksi haitatakseen laitteiston suoritusky-

kyä. Testattaessa vikatilanteiden havaitsemista korotettiin hälyttävän käyttäytymisen rajoja prosessorikuormituksen ja vapaan käyttömuistin osalta ja seurattiin testausympäristön laitteistoja kuormitettaessa valvontasovelluksen reagointia.

### 7.3.3 Vikatilanteen havaitseminen valvottavassa aliverkossa

Valvottavan palvelinympäristön kaikilta komponenteilta haetaan tilatiedot valvontaverkon ylimmälle tasolle käsiteltäviksi ja nämä käsitellään mahdollisten vikatilanteiden havaitsemiseksi, jonka testaamista edellisessä luvussa 7.3.2 testattiin. Valvottavassa ympäristössä on mahdollista esiintyä kuitenkin myös tilanteita, jolloin valvontatietoja ei voida tulkita niiden puuttumisen vuoksi esimerkiksi tietoverkko-ongelmien johdosta. Vikaantumisen havaitseminen valvottavan ympäristön alikomponenteissa testattiin estämällä verkkoyhteys alikomponenteille.

### 7.3.4 Vikatilanteen havaitseminen ylemmän tason komponentissa

Valvottavan palvelinympäristön jokaisella yksittäisellä komponentilla on mahdollisuus vikaantua ja tämä voi kohdistua myös yksittäisen komponentin tilatietoja keräävään ylemmän tason komponenttiin. Toimivan valvontasovelluksen aikaansaamiseksi tällaisessa tilanteessa alemman tason komponentin tulee kyetä havaitsemaan ylemmällä tasolla oleva potentiaalinen ongelma. Tämän testaamiseksi valvontasovelluksen ylemmän tason komponentilta katkaistiin verkkoyhteydet ja tämän jälkeen seurattiin alemman tason komponenttien reagointia ja valvontaverkon korjausoperaatiota, joiden avulla valvontatietojen käsittely voidaan taata suurimmassa osassa valvottavaa palvelinympäristöä.

## 7.4 Yhteenveto

Valvontasovelluksen toteutuksen toimivuutta testattiin niin yksittäisiin komponentteihin kohdistuneen moduulitestauksen, moduulien välisien rajapintojen toimintaan keskittyvän integraatiotestauksen, kuin järjestelmän yleiseen toimivuuteen keskittyvän järjestelmätestauksen avulla. Testauksen tarkoituksena oli varmentaa käytettyjen toteutustapojen toimivuus käytännössä ja todentaa sovelluksen vastaavan sille määrittelyvaiheessa asetettuja vaatimuksia.

# Luku 8

# Testauksen tarkastelu

Tämän luvun tarkoituksena on koostaa niitä havaintoja ja kokemuksia, jotka valvontasovelluksen testauksesta saatiin kerättyä.

Testien suorittaminen aiemmin esitellyn V-mallin mukaisesti aloittaen moduulitestauksesta selkeyttivät ohjelman toteutusta, koska seuraavaan toteutusvaiheeseen siirryttiin vasta aiemman osion läpäistessä testitapaukset. Tällaisella järjestelyllä ehkäistiin myös odottamattomien ongelmien ilmaantumista myöhäisemmissä toteutuksen vaiheissa. Haikalan ja Märijärven varoituksista huolimatta [5] testaajan ollessa myös ohjelmiston toteuttaja testitapaukset keskittyivät pääosin ohjelman toimivuuden varmentamiseen kuin virheiden varsinaiseen etsintään.

## 8.1 Sovelluksen ongelmakohdat

Sovellusta testattaessa havaittiin sovelluksen toimivuutta tai käyttöä haittaavia ominaisuuksia, joita on selvitetty seuraavassa.

### 8.1.1 Komponenttien dynaamisen lisäämisen puuttuminen

Valvontasovelluksen testiympäristön konfiguraatio luetaan tiedostosta ohjelmaa käynnistettäessä ja tämä aiheuttaa ylimääräisiä haasteita tapauksessa, jossa valvottavaan palvelinympäristöön ollaan lisäämässä uusia komponentteja. Nykyinen järjestely pakottaa ylläpidon lisäämään uuden komponentin tunnisteen ja sijainnin jokaiselle verkon komponentille erikseen. Tämä on työläs operaatio laajassa palvelinympäristössä ja jatkokehityksenä on erittäin suositeltavaa toteuttaa muunneltu versio verkon vikatilanteen muutosviestistä, jonka voisi lähettää verkon pääsolmun hallintaliittymästä ja jonka tehtävänä on ilmoittaa verkon uudesta komponentista.

### 8.1.2 Hälytysrajojen konfiguroinnin hankaluus

Valvontasovelluksen ytimen ollessa sangen joustava erilaisten valvottavien suureiden suhteen havaittiin laitteiston konfiguroinnin kannalta haasteelliseksi

määrittää kullekin suureelle hälytysrajat häiriötapauksia varten. Testauksessa tyydyttiin järjestelyyn, joka tulkitsi valvottavan suureen nimen perusteella kategorian, jonka mukaan hälytysrajat asetettiin. Tällainen järjestely ei välttämättä kuitenkaan tyydytä monimutkaisemman ja laajemman palvelinympäristön ylläpitoa, vaan hälytysrajat tulisi voida välittää jotenkin valvovien komponenttien kesken dynaamisesti.

### 8.1.3 Laitealustan vaikutus suorituskykyyn

Valvontasovellus toteutettiin Javalla siten, että se suunniteltiin toimimaan JBoss J2EE-sovelluspalvelimella. Testiympäristössä havaittiin tämän sovelluspalvelinehdon aiheuttavan ongelmia vanhemman sukupolven laitealustoilla. Siinä, missä AMD64 suoritinytimellä varustettu laitteisto suoriutui kuormituksesta suorituskyvyn heikentymättä, oli Pentium III suoritinytimellä varustetun laitteiston hankala suoriutua edes sovelluspalvelimen ajamisesta. Tämä asettaa määrittelyssä asetetun vaatimuksen, että valvontasovellus ei saa aiheuttaa häiriötä itse palvelusovelluksille, kyseenalaiseksi ja näinollen on perusteltua paneutua jatkokehityksessä myös etsimään valvontasovelluksen sovelluspalvelinalustaksi kevyempää vaihtoehtoa.

# Luku 9

# Yhteenveto

Tietojärjestelmistä on tullut yhä korvaamattomampia yhteiskunnan peruspilareita tietomäärien alati kasvaessa. Yrityksien ja yhteisöjen tietojärjestelmien määrä on kasvanut ja tärkeää on näiden järjestelmien toimivuuden takaaminen. Tietojärjestelmien monimutkaistuessa näiden toimintakunnon varmistaminen ja mahdollisten ongelmatilanteiden havaitseminen ja selvittäminen on ylläpidollisesti hankaloitunut. Tämän lopputyön tarkoituksena oli tutkia mahdollisuuksia tehostaa ja selkeyttää ylläpidon hajautetusta tietojärjestelmästä saamaa informaatiota ja nopeuttaa ongelmakohtien paikantamista.

Lopputyössä pyrittiin kehittämään valvontasovellus, jota voitaisiin käyttää valvontatiedon keräämiseen tietojärjestelmän toteutusrakenteesta riippumatta siten, että valvottavat suureet voitaisiin konfiguroida vapaasti. Valvontasovelluksella tuli olla myös mahdollisuus nähdä valvontatietojen kehittyminen pidemmällä aikavälillä ja informoida ylläpitoa havaituista ongelmatilanteista tietojärjestelmän ongelmatilanteista huolimatta.

Valvontasovellus toteutettiin Javalla ja tietojärjestelmän laitealustakohtaiset valvottavat tiedot päädyttiin keräämään laitealustakohtaisesti konfiguroitavissa olevin komentotulkkiskriptein. Hajautetun tietojärjestelmän valvontatiedon koottu tulkitseminen päädyttiin toteuttamaan hierarkisella tähtiverkkorakenteella, jotta valvontasovelluksen aiheuttama verkkokuormitus voitaisiin pitää tietojärjestelmän varsinaista toiminnallisuutta häiritsemättömänä.

Valvontasovelluksen dynaamisesti ongelmatilanteissa muuttuva rakenne osoittautui haasteelliseksi ongelmaksi ratkaista, sillä vaikka valvontasovellus kykenee lähettämään hälytysinformaatiota ylläpidolle, ei ylläpito nopeassa tilanteessa tiedä, mikä tietojärjestelmän osa kulloinkin valvontasovelluksen pääsolmua suorittaa tietojen tarkistamiseksi. Reaaliaikaisen valvontaverkkohierarkian esittämistä tulisi näinollen tarkentaa käyttöliittymässä.

Resurssipulan vuoksi valvontasovelluksen käyttöliittymä keskittyy esittämään ainoastaan reaaliaikaista näkymää valvottaviin komponenttikohtaisiin suureisiin ja näinollen valvontasovelluksen valvontatietojen kehitysseurannan toteutus rajattiin pois testiympäristöä toteutettaessa. Valvottavasta tietojärjestelmästä kerättyjen tietojen perusteella toteutettava tulevien ongelmatilan-

teiden ennustaminen päätettiin myös rajata pois aikataulullisista ongelmista.

Sovelluksen toteutuksen ja testauksen yhteydessä havaittiin toteutusmalli idealtaan toimivaksi, joskin verkon dynaaminen konfiguroiminen testiympäristössä sekä eri suureiden kriittisen käyttäytymisen havainnointi osoittautui haastavaksi ongelmaksi. Vaikka valvontasovellus on toteutustekniikaltaan joustava ja tilatietoja voidaan testien perusteella siirtää salattuina myös julkisessa Internet-verkossa, on valvontasovellusta haasteellista saada toimimaan useamman erillisen sisäverkon tapauksessa. Valvontasovellus antaa mahdollisuuksia monen tyyppiselle tietojärjestelmän seurannalle kerätessään valvontatietoja pidemmältä aikaväliltä, sen avulla valvottavan tietojärjestelmän tiedot voidaan koostaa selkeämpään muotoon ja näiden tietojen avulla ylläpidon on mahdollista keskittyä ylläpidollisesti tärkeisiin osiin tietojärjestelmän toiminnan takaamisessa.

# Luku 10

# Tulevaisuus

Hajautettujen tietojärjestelmien määrä tulee kasvamaan tehtäväalueiden monimutkaistuessa ja yhteisön vaatiessa järjestelmiltä parempaa suorituskykyä ja parempaa toimintavarmuutta. Ylläpidollisesta näkökulmasta valvottava ympäristö laajenee ja valvontainformaatio hajautuu erilaisien järjestelmien johdosta vaikeasti hallittavaksi.

National Instruments toteaa hajautettujen valvontajärjestelmien toteuttamisesta, että on hyväksi pyrkiä valvonnassa mahdollisimman suureen automaation asteeseen ja että avoimella, joustavalla toteutuksella mataloitetaan kynnystä liittää eri valmistajien tuotteita samaiseen kokonaisuuteen, jolloin kokonaisuus saadaan entistäkin tehokkaammaksi [1]. Näin tarkasteltaessa toteutettavana olleelle geneeriselle valvontasovellukselle sovelluskohteiden löytyminen ei tuottane ongelmia jatkossa.

Lopputyössä toteutetulla valvontasovelluksella, joka voitaisiin konfiguroida vastaanottamaan sekä prosessoimaan valvontatietoa lähteestä riippumatta, tulee olemaan käyttökohteita tulevaisuuden ylläpitoyhteisöissä. Tämän hyödyntäminen tulevaisuuden hajautettujen tietojärjestelmien ja palvelinympäristöjen valvonnassa vaatii kuitenkin sovelluksen konfiguroinnin yksinkertaistamista, käyttöliittymän kehittämistä sekä valvottavan ympäristön statistiikan tarkastelemisen parantamista ympäristön käyttäytymisen kattavan tulkitsemisen mahdollistamiseksi. Näiden haastavien kehityskohteiden selvittämisen jälkeen tällaisella valvontasovelluksella voidaan yksinkertaistaa erinäisten tietojärjestelmien valvontaa ja useita toisistaan erillisiä järjestelmiä voidaan yhdistää saman valvontaympäristöön kuuluviksi. Valvontaa pystytään tämän sovelluksen avulla keskittämään sekä ylläpitotyöt voidaan kohdistaa kriittisiin valvontaympäristön osakokonaisuuksiin kokonaisuuden silti tästä hämärtymättä.

# Kirjallisuutta


[1] *Building Distributed Monitoring and Control Systems with LabVIEW*, 2005. URL http://zone.ni.com/devzone/conceptd.nsf/webmain/3FE99D648E501%53786256C8400703ABD.

[2] Ahola, Jero. *Tiedonkeruujärjestelmä teollisuussähköjärjestelmien tiedonhallintaa varten.* pro gradu. URL http://www.ee.lut.fi/stats/estat_logger.php?ID=138.

[3] Cristian, Flaviu, Dancey, Bob & Dehn, Jon. *Fault-tolerance in air traffic control systems. ACM Trans. Comput. Syst.*, osa 14(3):265–286, 1996. ISSN 0734-2071.

[4] Devanbu, Premkumar T. & Stubblebine, Stuart G. *Software engineering for security: a roadmap.* Teoksessa *ICSE - Future of SE Track*, sivut 227–239, 2000. URL citeseer.ist.psu.edu/devanbu00software.html.

[5] Haikala, Ilkka & Märijärvi, Jukka. *Ohjelmistotuotanto.* Suomen Atk-kustannus Oy, PL 325,00181 HELSINKI, 6 painos, 1998. ISBN 951-762-696-7.

[6] Hairo, Mika. *Internet-pohjaisen liiketoimintasovelluksen käyttökatkon aiheuttamien kustannusten arviointi.* Diplomityö, Espoo, 2004.

[7] Kaikkonen, Antti. *Availability Performance of Semi-automated Production Platform: Analysis and Improvement.* Diplomityö, Espoo, 2005.

[8] Kalliala, Eija. *Laatu tietojärjestelmän ylläpitotyössä..* Ylemmän vakuutustutkinnon tutkielma nro 333, Vakuutusalan koulutuskeskus, Helsinki, 1995.

[9] Koistinen, Heikki. *Tietojärjestelmien ylläpito.* Talentum Media Oy, 2002. ISBN 951-762-809-9.

[10] Lahtela, Marko. *Entity Bean-komponentit Java 2 Platform Enterprise Edition-arkkitehtuurissa.* Helsingin liiketalouden ammattikorkeakoulun opinnäytetyö, Helsingin Liiketalouden Ammattikorkeakoulu, Rautatieläisenkatu 5, FIN-00520 Helsinki, 2003.

[11] Leppinen, Timo, Hirvensalo, Jorma & Maksimainen, Väinö. *Tietoliikennealan ohjelmistojen ylläpito.* Tiedotteita, 141, Valtion teknillinen tutkimuskeskus, Teletekniikan laboratorio, Espoo, syyskuu 1982.

[12] Mattila, Ari. *Tietojärjestelmän käyttöönotto*. Diplomityö, Espoo, 1999.

[13] M.Crichlow, Joel. *Hajautetut tietojärjestelmät*. Edita Oyj, IT Press PL 760,00043 EDITA, 2001. ISBN 951-826-544-5.

[14] Ruini, Henri. *Johdatus XML-tekniikkaan*, 2001. URL `http://www.cs.helsinki.fi/u/ruini/structure/xml/`.

[15] Smith, Curtis & Henry, David. *High-performance Linux cluster monitoring using Java*, 2002. URL `www.linuxclustersinstitute.org/Linux-HPC-Revolution/Archive/P%DF02/32-Smith_C.pdf`.

[16] Takki, Pekka. *IT-sopimukset käytännön käsikirja*. Talentum Media Oy, 2 painos, 2002. ISBN 952-14-0644-5.

[17] Tilastokeskus. *Tiedolla tietoyhteiskuntaan*. Tilastokeskus, 4 painos, 2003. ISBN 952-467-224-3.

[18] Tilastokeskus. *Tietoyhteiskunta*, 2004. URL `http://www.stat.fi/tk/yr/tietoyhteiskunta/`.

[19] Turban, Eifram, McLean, Ephraim & Wetherbe, andmes. *Information technology for management: transforming business in the digital economy*. John Wiley & Sons, inc., 3 painos, 2002. ISBN 0-471-40075-0.

[20] Valtionvarainministeriö. *Tietoturvasanasto*, 2000. URL `http://www.vm.fi/tietoturvasanasto/sisallys.htm`.

[21] Viestintävirasto. *Tietoturva:Tietoturvallisuuden perusteet*, 2004. URL `http://www.ficora.fi/suomi/tietoturva/ttperusteet.htm`.

[22] White, Karen S., Areti, Hari & Garza, Omar. *Control System Reliability at Jefferson Lab*. Teoksessa *ICALEPCS these Proceedings.*. URL `citeseer.ist.psu.edu/422194.html`.

[23] Ylilammi, Matti J., Bergman, Kimmo, Honkasaari, Terttu, Hälikkä, Timo & Jalasoja, Kirsti. *Ylläpito: Tutkimus suomalaisten tietojärjestelmien ylläpidosta*. Tutkimusraportti a:15, Tietotekniikan kehittämiskeskus ry, Helsinki, 1988.